# RETURN FROM THE ANT
## SYNTHETIC ECOSYSTEMS FOR MANUFACTURING CONTROL

## DISSERTATION

zur Erlangung des akademischen Grades
doctor rerum naturalium
(Dr. rer. nat.)
im Fach Informatik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät II
Humboldt-Universität zu Berlin

von
Herr Dipl.-Inf Sven Brückner
geboren am 04.10.1971 in Bad Saarow-Pieskow

Präsident der Humboldt-Universität zu Berlin:
Prof. Dr. Dr. h.c. Hans Meyer

Dekan der Mathematisch-Naturwissenschaftlichen Fakultät II:
Prof. Dr. sc. nat. Bodo Krause

Gutachter:

1. Prof. Dr. Hans Dieter Burkhard
2. Priv.-Doz. Dr. Kurt Sundermeyer
3. Van Dyke Parunak, Ph.D.

eingereicht: 22.01.2000
Tag der mündlichen Prüfung: 21.06.2000

**Zusammenfassung**

Die vorliegende Dissertation hat einen technologischen und einen anwendungsbezogenen Schwerpunkt. Technologisch ordnen sich die präsentierten Forschungsergebnisse in das Gebiet der "Swarm Intelligence" (dt.: Schwarm-Intelligenz) ein. Swarm Intelligence ist ein Teilbereich der Informatik, der sich an der Überschneidung zwischen der Multi-Agenten Systeme Forschung der Künstlichen Intelligenz und dem Forschungsgebiet "Artificial Life" (dt.: Künstliches Leben) befindet. Im Gegensatz zur Swarm Intelligence im allgemeinen, überträgt der spezielle Ansatz "Synthetic Ecosystems" (dt.: synthetische Ökosysteme) nicht nur Koordinationsmechanismen aus biologischen Multi-Agenten Systemen, wie zum Beispiel Insekten Kolonien, in den Entwurf künstlicher Systeme. Vielmehr sollen die grundlegenden Prinzipien "natürlich" entstandener komplexer Systeme, also auch zum Beispiel einer Aktienbörse, übernommen werden.

Als anwendungsbezogener Hintergrund der Dissertation wurde die verteilte Steuerung moderner industrieller Fertigungsanlagen gewählt. Die Fertigungssteuerung ist ein geeignetes Anwendungsfeld für die Technologien, die im Rahmen der Forschungsarbeiten entwickelt wurden. Damit dient die Präsentation eines synthetischen Ökosystems für die Fertigungssteuerung der Demonstration des neuartigen Ansatzes zum Entwurf, Realisierung und Evaluierung komplexer, industriell relevanter Systeme. Gleichzeitig leistet die vorgestellte Architektur der Fertigungssteuerung und die darin verwandten Koordinationsverfahren einen Beitrag zur Weiterentwicklung holonischer Produktionssysteme. Der holonische Ansatz zur Produktionsplanung und -steuerung genießt derzeit große Aufmerksamkeit sowohl in der Forschung als auch in der Industrie. Als Teilgebiet der Entwicklung intelligenter Fertigungssysteme (engl.: IMS – Intelligent Manufacturing Systems), propagiert der holonische Ansatz eine Abkehr von der traditionell zentralistischen und hierarchischen Planung und Steuerung hin zu selbst-organisierenden Systemen autonom (inter-)agierender Individuen ("Holone"). Bei der praktischen Umsetzung holonischer Systeme werden sehr häufig Technologien aus der Multi-Agenten Systeme Forschung angewandt. Mit dieser Dissertation rücken auch synthetische Ökosysteme in das Blickfeld holonischer Systeme.

Natürliche Agentensysteme im allgemeinen und Kolonien sozialer Insekten im besonderen faszinieren durch ihre Robustheit, ihre Flexibilität und ihre Anpassungsfähigkeit. Solche Systeme bestehen häufig aus sehr vielen, sehr einfachen Individuen und doch weisen sie ein komplexes und koordiniertes Gesamtverhalten auf. Es gibt mehrere Zweige in unterschiedlichen Wissenschaften, zum Beispiel in der Biologie, Physik, Ökonomie oder in der Informatik, die sich mit verteilten Systemen lokal interagierender Individuen beschäftigen. Ihre Erforschung resultiert in einer Reihe wiederholt beobachteter grundlegender Eigenschaften. Um künstlich erschaffene Systeme mit ähnlichen Eigenschaften auszustatten werden Entwurfsprinzipien für das Design von Multi-Agenten Systemen in dieser Dissertation vorgeschlagen. Jedes Entwurfsprinzip wird systematisch eingeführt, motiviert und in seinen Konsequenzen für Anwendungen in der Fertigungssteuerung diskutiert.

Stigmergie ist ein grundlegendes Konzept der Koordination einer großen Anzahl von Individuen unter anderem in Kolonien sozialer Insekten. Die Formulierung dieses Konzepts ist auf den Biologen Grassè zurückzuführen, welcher in der Mitte des zwanzigsten Jahrhunderts das Schwarmverhalten von Termiten untersuchte. Stigmergie beruht auf der Tatsache, daß das Verhalten eines jeden Individuums durch die aktuelle Konfiguration seiner lokalen Umwelt bestimmt wird. Die Umwelt wiederum, wird durch die Aktivitäten der Individuen verändert. Diese Wechselwirkung führt in Verbindung mit

entsprechend ausgelegten individuellen Verhaltensmustern zur Emergenz einer global koordinierten Erfüllung der anstehenden Aufgaben der Kolonie. Im Detail wird sematektonische von marker-basierter Stigmergie unterschieden, wobei bei sematektonischer Stigmergie der Zustand der Aufgabenerfüllung selbst (z.B. Stand des Nestbaus) das Individualverhalten beeinflußt, während marker-basierte Stigmergie aufgabenunabhängige Marker (z.B. Pheromone) in der Umwelt platziert.

Multi-Agenten Systeme finden ihre Realisierung in Software, welche gegebenenfalls an physische Aktuatoren gekoppelt ist. Im allgemeinen besteht diese Software aus einer Laufzeitumgebung und den darin ausgeführten Agenten. Die vorliegende Dissertation präsentiert eine Erweiterung von Laufzeitumgebungen um eine anwendungsunabhängige Pheromon Infrastruktur (*PI*). Die *PI* ermöglicht es den Softwareagenten des jeweiligen synthetischen Ökosystems, künstliche Pheromone als Datenstrukturen in einem virtuellen Raum abzulegen und wahrzunehmen. Diese Datenstrukturen dienen als Marker in stigmergetischen Koordinationsmechanismen. Die Algorithmen der *PI* operieren auf diesen künstlichen Pheromonen und emulieren die natürlichen Vorgänge der räumlichen Ausbreitung und Verdunstung von Pheromonen auf abstrakter Ebene. Zusätzlich wird das natürliche Vorbild um eine automatische Aufbereitung von Informationen erweitert.

Die Funktionalität der *PI* wird in dieser Dissertation spezifiziert. Des weiteren wird ein formales Modell erstellt, welches die Grundlage einer numerischen Analyse der Eigenschaften der *PI* bildet. Die Analyse liefert Vorhersagen für das Entstehen von räumlichen Mustern von Pheromonkonzentrationen in der *PI*. Diese Vorhersagen können dann in der Feineinstellung und der Evaluierung von Koordinationsmechanismen verwendet werden. Außerdem dient das formale Modell als Grundlage für den Beweis der globalen Stabilität der *PI*. Damit ist gesichert, daß unabhängig von der gewählten räumlichen Struktur und den von der jeweiligen Anwendung generierten Pheromonen die Konzentrationen der Pheromone immer in ihrer Stärke begrenzt sind. Der Beweis der globalen Stabilität ist eine wichtige Voraussetzung für die Verwendung der *PI* in praktischen Anwendungen.

Die Spezifikation einer verteilten Realisierung der *PI* bildet den Abschluß der allgemeinen Betrachtung. Die Agenten, welche die (virtuelle) räumliche Struktur der *PI* widerspiegeln, werden im Detail spezifiziert. Auf der Basis dieser Spezifikation ist im Rahmen der Dissertation ein Prototyp der *PI* realisiert worden. Dieser Prototyp diente dem Nachweis des vorhergesagten Verhaltens der Infrastruktur und der späteren Evaluierung des entwickelten Fertigungssteuerungssystems.

Im weiteren Verlauf der vorliegenden Dissertation wird ein neuartiger Ansatz zur Fertigungssteuerung betrachtet. Die absehbaren Veränderungen der äußeren Bedingungen der industriellen Produktion, ausgelöst durch den globalen Übergang von Anbieter- zu Verbrauchermärkten, erfordert die Fertigung immer komplexerer und variantenreicherer Produkte in ständig schwankenden Stückzahlen und deutlich verkürzten Lebenszyklen bei gleichzeitig sinkenden Kosten. Zur Erfüllung dieser Anforderungen in der Massenproduktion wandelt sich die traditionell starr verkettete Strangfertigung (z.B. Transferstraßen) zur flexiblen Fließfertigung (z.B. flexible Bearbeitungszentren). Die Steuerung einer flexiblen Fließfertigung erfordert neue Herangehensweisen. In einer holonischen Fertigung, zum Beispiel, organisiert sich die Produktionsplanung und Produktionssteuerung selbst um die Erfüllung der aktuellen Aufträge. Dabei werden in der Steuerung verteilte, reaktive Verfahren verwendet, welche eine deutlich gesteigerte Robustheit und Flexibilität gegenüber Störungen und Veränderungen aufweisen.

Der Übergang zur flexiblen Fließfertigung bedeutet die Einführung von Flexibilität in

der Bearbeitung aber auch im Transport des Materials. Es ist eine grundlegende Eigenschaft dieser Fertigungssysteme, daß zu einem beliebigen Zeitpunkt eine Reihe möglicher Transportwege und damit eine Vielzahl möglicher Muster im Materialfluß zur Verfügung stehen. Dabei führt aber nur eine kleine Menge dieser Muster zu einer bestmöglichen Erfüllung der globalen Produktionsziele (z.B. hoher globaler Durchsatz). Es ist also die Aufgabe der Fertigungssteuerung in jeder Situation das bestmögliche Materialflußmuster zu erreichen. Ist ein verteilter Ansatz für die Steuerung gewählt worden, so muß diese Optimierung nach globalen Produktionszielen in die lokalen Steuerungsentscheidungen integriert werden, ohne die Autonomie der lokalen Einheiten zu verletzen.

Die Dissertation präsentiert ein sogenanntes geführtes Fertigungssteuerungssystem (*GFSS*), welches einen verteilten und reaktiven Steuerungsansatz mit einer Flußoptimierung unter Beachtung globaler Produktionsziele in neuartiger Weise verbindet. Der Entwurf des *GFSS* folgte den vorgeschlagenen Prinzipien für synthetische Ökosysteme und die Agenten im *GFSS* werden mit Hilfe der Pheromon Infrastruktur koordiniert. Die Agenten und Pheromone des *GFSS* werden detailliert spezifiziert und in einem realistischen Beispiel aus der Automobilindustrie evaluiert. In der Evaluierung wird von den Ergebnissen der Analyse der *PI* Gebrauch gemacht. Die dabei gewählte numerische Beschreibung des Einzelverhaltens und die darauf aufbauende Betrachtung des emergierenden Gesamtverhaltens weist den Weg zu einer systematischen Evaluierung von emergenten Systemeigenschaften in synthetischen Ökosystemen.

In einem abschließenden Kapitel werden die drei inhaltlichen Schwerpunkte der Dissertation noch einmal betrachtet. Vor dem Hintergrund des *GFSS* werden die vorgeschlagenen Entwurfsprinzipien für synthetische Ökosysteme systematisch auf ihre Anwendbarkeit und praktische Bedeutung hin überprüft. Außerdem wird die allgemeine Verwendung der *PI* für den Austausch von Informationen zwischen Agenten untersucht. Und schließlich wird die Fertigungssteuerung aus der Sicht abstrakter Zustandsräume diskutiert.

Die vorliegende Dissertation weist den Weg für eine Reihe weiterführender Forschungsarbeiten. So werden zum einen detaillierte Konzepte für die Erweiterung des *GFSS* um eine automatische Strategiebewertung und -generierung und um ein Visualisierungssystem vorgestellt. Zum anderen werden aber auch notwendige Ergänzungen der Entwurfsprinzipien und mögliche Verbesserungen der *PI* und des darauf basierenden Evaluierungsansatzes vorgeschlagen.

Schlagwörter:

Multi-Agenten Systeme
Produktionssteuerung
Koordination
Stigmergie
Swarm Intelligence
Emergenz
Selbst-Organisation

**Abstract**

The synthetic ecosystems approach attempts to adopt basic principles of natural ecosystems in the design of multiagent systems. Natural agent systems like insect colonies are fascinating in that they are robust, flexible, and adaptive. Made up of millions of very simple entities, these systems express a highly complex and coordinated global behavior.

There are several branches in different sciences, for instance in biology, physics, economics, or in computer science, that focus on distributed systems of locally interacting entities. Their research yields a number of commonly observed characteristics. To supply engineered systems with similar characteristics this thesis proposes a set of principles that should be observed when designing synthetic ecosystems. Each principle is systematically stated and motivated, and its consequences for the manufacturing control domain are discussed.

Stigmergy has shown its usefulness in the coordination of large crowds of agents in a synthetic ecosystem. Sign-based stigmergy through synthetic pheromones is supported by an extension to runtime environments for software agents called the pheromone infrastructure. In this thesis the operation of the pheromone infrastructure is specified, formally modeled and analyzed, and an implementation is presented.

The guided manufacturing control system for flexible flow shops is designed following the proposed principles and it uses the pheromone infrastructure to coordinate its agents. It comprises two subsystems. The control (sub)system, which enables production, is distributed and reactive. The advisory (sub)system observes the operation of the control system and advises the manufacturing execution under global considerations. This thesis specifies the guided manufacturing control system and evaluates its operation in a simple but realistic example adapted from the automotive industry. The applicability of the design principles, the usage of the pheromone infrastructure, and the operation of manufacturing control in abstract state spaces are considered on the basis of the guided manufacturing control system.

Keywords:

Multi-Agent Systems
Manufacturing Control
Coordination
Stigmergy
Swarm Intelligence
Emergence
Self-Organization

Contents

## Aknowledgements

At the end there is only my name showing on the cover of the thesis, claiming all faults of this work as mine. As for the contributions – research is not done in thin air and so there are many people I would like to thank here.

I would like to thank my supervisor and long-time supporter during my time at Humboldt University, Professor Hans-Dieter Burkhard. From early on when I just had my first encounters with Artificial Intelligence and Multi-Agent Systems he guided me to develop my own ideas and he challenged me to make them work. At the same time he helped me to achieve the scientific rigor required for a dissertation.

Thanks are also due to my second supervisor Dr. Kurt Sundermeyer at the DaimlerChrysler AG Research & Technology laboratories in Berlin. He gave me the opportunity to understand the complexity and the requirements of industrial manufacturing processes and then his trust encuraged me to leave conventional approaches behind and explore new paths.

The scientific work of Dr. Van Dyke Parunak had been one of my first inspirations when I decided on the subject of my dissertation research and so I was very happy to gain his support when we finally met. Since then we had many fruitful discussions and he helped me with his deep insights into the subject but also with his suggestions for the improvement of the style and the language of the dissertation.

Scientific research is teamwork and without my friends and colleaguages in the Multi-Agent Systems research department of DaimlerChrysler I would not have been able to write this dissertation. Together we created an interested, open-minded, and challenging environment that allows ideas to grow and to interact. So, I would like to thank Stefan Bussman, Dr. Klaus Schild, Dr. Afsaneh Haddadi, and especially Dr. Hartwig Baumgärtel who even believed in me when I did not.

Last, but definietly not least, there is my family with my parents Schlomith and Frank and my sister Aviva to thank for being there whenever I needed understanding and encuragement. They helped me to stay in touch with the real world at times when my head was up in the clouds.

**Abbreviations**

| | |
|---|---|
| SE | Synthetic Ecosystem |
| MAS | Multi-Agent System |
| PI | Pheromone Infrastructure |
| GMC | Guided Manufacturing Control |
| $P_A$ | „Ability“ Pheromone |
| $P_S$ | „State“ Pheromone |
| $P_P$ | „Prediction“ Pheromone |
| $P_F$ | „Force“ Pheromone |
| $P_T$ | „Transition“ Pheromone |

# Chapter 1 - Introduction

*The following chapter motivates the research that led up to this thesis. It sets the goal of the thesis, which is put into four questions. Finally, the structure of the thesis is presented, introducing the following chapters and major sections.*

## 1.1 Introduction

### 1.1.1 Motivation and Background

In summer 1998, there arose the following task in the ESPRIT LTR project MASCADA: Given a segment of a transport system of arbitrary layout in discrete high-volume production composed of unidirectional line-buffers (e.g., conveyors) and multi-input multi-output sequential routing devices (e.g., rotation tables, lifts), and assuming that the workpieces sent through the segment are all of one product but may be differentiated on the basis of the value of one product parameter; how may the segment be controlled in a decentralized manner so that the outflow of workpieces occurs in batches of workpieces of the same product parameter value with the average batch size of the outflow being significantly higher than that of the inflow?

Businesses around the globe face a change from a supplier's market to a customer's market. As one consequence of the change flexibility makes its way into high-volume production where sequential and optimized transfer-lines previously dominated. The introduction of flexibility in products and processes requires a change in the manufacturing system as well as in its control. Traditionally, manufacturing control is organized top down. Strategical planning passes its results down to tactical planning, which in turn triggers operational planning processes. The final result of the planning process, a detailed schedule, is given to the manufacturing execution to be implemented on the shop floor. There, unforeseen disturbances are encountered that may invalidate the schedule and thus trigger a new planning cycle at the operational level.

Modern flexible flow shops require a new approach to control, one that self-organizes around the manufacturing execution. The batching problem stated at the beginning cannot be solved by a planning-scheduling-execution cycle. It is highly dynamic and it requires an ongoing concern rather than a one-time achievement. At any time, new workpieces may enter the system while others leave it. The volume of the inflow and the mix of the variants fluctuate strongly over time. It is not even known how many different variants have to be handled at a time. In addition to such short-term changes, there are changes or disturbances in the layout of the transport system. Finally, even the criteria the workpieces are sorted by may be subject to long-term change.

Social insect colonies are an example of distributed systems of locally interacting individuals. They display a wide variety of system-level properties that are also required of modern manufacturing systems and their control. Emerging from simple and often indirect interactions of individuals that are simple compared to the complexity of the system, insect colony behavior is robust, flexible, adaptive, self-organizing, intuitive, and scalable.

In recent years interdisciplinary research has focused on insect colonies and similar

systems in physics, chemistry, economics, biology, or computer science. The research follows two major approaches: the analytic approach and the engineering approach. The analytic approach takes individual behavior and a specification of the interactions and seeks to determine what system-level properties emerge. The engineering approach starts out with a set of system level properties and asks what individuals and what interactions are required to achieve these properties.

Taking the inspiration from insect coordination mechanisms, the following simple but effective solution to the batching problem was found. Each sequential router is assigned a Router-agent. A Router-agent acts completely autonomous without even directly communicating with other agents. The simple task fulfilled by each Router-agent is to take workpieces sequentially from the entries of its router to the exits. Therefore, an agent needs to perceive the workpieces waiting at the entries and the current state of the exits (blocked or free). To achieve the required batching property at the level of the control system made up of these agents, a Router-agent has a simple memory. There it stores for each exit the value of the product parameter of the last workpiece that has passed the exit.

With multiple entries and multiple exits a Router-agent has to decide when to take which of the available workpieces to what exit. The decision is taken in a sequential execution of the following three simple rules, starting at rule one:

**Sorting Rule (1)**.—IF at entry X there is a workpiece with a product parameter value of p and there exists a free exit Y whose related product parameter in agent memory has a value of p too, THEN take the workpiece from X to Y immediately and restart at rule one.

> **Extension**.—IF there are currently more than one such actions possible, THEN select one of them randomly.

**Blocking Rule (2)**.—The ratio of free entries to the overall number of entries determines a probability $P_B$ to pause the routing operation for a fixed time $T_B$. The higher the ratio, the higher is the probability to pause. After pausing restart at rule one.

**Random Rule (3)**.—Select one occupied entry X and one free exit Y randomly, route the workpiece from X to Y, and then restart at rule one.

These rules are based on the assumption that every workpiece may be routed from any entry to any exit at each router. As a consequence, the following two requirements for the layout of the transport system are set:

**„Open“ Layout**.—In the segment of the transport system under consideration, every entering workpiece must be permitted to leave through any exit.

**„Directed“ Layout**.—There must be no cycles in any path from an entry to an exit of the segment.

There exist extensions to the Router-agent behavior that provide an explicit global routing in addition to the sorting of workpieces. But these extensions are outside of the scope of this introduction.

The inspiration for the design of the Router-agents came from the coordinated nest construction of ants, termites, or bees. The basic principle that governs the coordination is sematectonic stigmergy. In stigmergy in general, there are mechanisms that trigger individual work (Greek: „ergon“) through signs (Greek: „stigmata“) in the environment. If these signs are aspect of the task itself (e.g., form of an arc in a termites nest) then it is the sematectonic form of stigmergy. In sign-based stigmergy on the other hand, the task

fulfillment is coordinated through additional markers (e.g., pheromones). Stigmergy requires agents to act upon changes in their environment that are caused by the agents themselves. These repetitions in space and time of small-scale activities in the environment result in a stable and self-organized system-level behavior.

A Router-agent takes decisions on the current configuration of its local environment. Through its actions it changes not only its own environment, but also the environment of other downstream and upstream Router-agents is changed too. The emergence of coordinated system-level behavior (here batching) requires the individual activities to be repeated as often as possible. As a consequence the quality of the task fulfillment by the agent system depends on the individual behavior as well as on the structure of the stigmergetic interactions as it is given by the layout of the transport system. Good batching behavior emerges if the following requirements for the layout are fulfilled:

**„Alternative“ Layout**.—There are many possible paths from an entry to an exit of the segment and these paths should intersect as often as possible to provide a large number of local routing points.

**„Homogeneous“ Layout**.—Most sequential routers have the same number of entries and exits to permit a homogeneous setting of agent parameters.

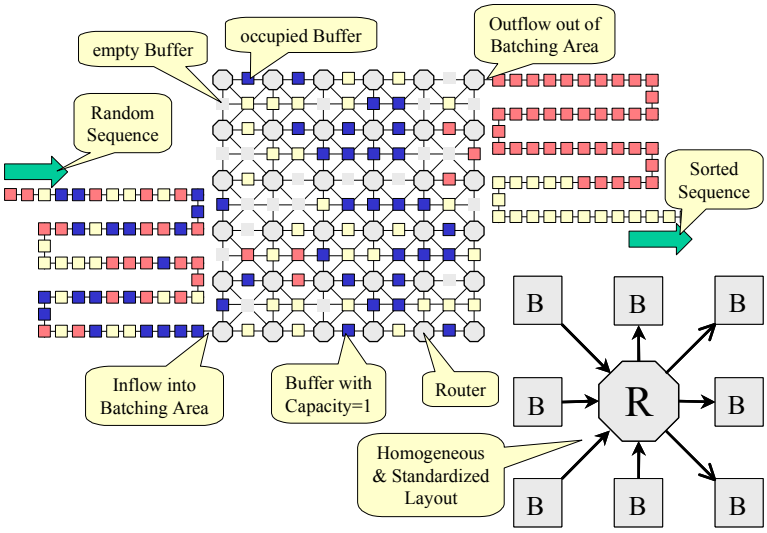


Figure 1.1. Layout of Router-agents with a High Batching Quality

The Router-agents have been implemented and they have proven themselves in several different layouts. One of the most effective layouts for batching is the matrix layout as it is shown in Figure 1.1. The illustrated segment has only one entry (lower left) and one exit (upper right). The different shades indicate different product parameter values. The distributed control system self-organizes to sort a random inflow into a high-quality outflow.

The design of a self-organizing control system that creates batches in an initially random material flow was at that time spontaneous and intuitive. But its success raised the question whether there is a systematic approach to the design. What underlying (bio-)logic has to be employed to successfully engineer an agent system of industrial strength that yields global properties comparable to those of complex natural systems? What specific support may be given to an engineer who eventually implements such an agent society? Finally, in addition to the engineering aspect, the analytic aspect also comes into play. Is there a way to support the tuning and the evaluation of coordination mechanisms that give

rise to emerging global properties?

Insect societies and the coordination mechanisms they employ have not been designed. They are the result of million years of evolution. In the course of evolution the individual behavior and the interactions had been selected to yield an optimal colony behavior. It is not the single insect behavior that is evaluated for its fitness because the individual cannot survive without the colony. The „goal“ of the optimization process is to achieve the best system-level properties in the given environment by tuning the individual behavior only. The ordering force of self-organization supports the process [Kauffman, 1995].

The engineering of agent systems follows the same goal – the resulting system is evaluated by its global properties. Hence, engineering could also try to design all required properties into a homogeneous set of agents, employing, for instance, artificial evolution. But more intuitive, and hence easier to realize and to change, is it to split the system into clusters of different agents, each cluster providing different aspects in the overall system-level behavior.

When considering the general requirements for modern manufacturing, two different kinds of system-level properties are identified. Primarily, the operation of the manufacturing system must be robust, agile, and flexible in the face of changes and disturbances. But, when these primary goals are reached, the system is also required to fulfill the production goals as good as possible. The clustering in the design of an agent system for manufacturing control may occur following such a distinction of properties. Therefore, the following question is raised: How should one design the interplay between the agents that achieve robustness and flexibility and those that seek to optimize the operation according to external production goals?

### 1.1.2 Goal and Structure of the Thesis

The work presented in this thesis aimed to answer the following four questions:

**Design**.—What principles should be followed if a distributed system of locally interacting individuals is designed so that it yields global properties like robustness, flexibility, agility, scalability, or intuitiveness?

**Realization**.—How may the services of a runtime environment that executes software-agents be extended so that it supports general sign-based stigmergetic interactions?

**Evaluation**.—Is there a formal approach to the prediction, the tuning, and the evaluation of stigmergetic multi-agent coordination mechanisms?

**Application**.—Given a set of design principles, an extended agent runtime environment, and a formalism for sign-based stigmergy, how is a self-organizing system for manufacturing control designed, tuned, and evaluated that combines robustness and flexibility with optimization according to production goals?

The thesis is structured as follows. Chapter 2 provides the context into which the work is set. Section 2.1 reviews concepts and techniques related to the interdisciplinary research into distributed systems of locally interacting individuals, while Section 2.2 presents the chosen application domain of manufacturing control with its requirements and approaches.

The Chapters 3, 4, 5, and 6 present the main contribution of the thesis. In Section 3.1 an extensive set of design principles are stated, motivated, and discussed in their implications for manufacturing control. To support the design of stigmergetic agent coordination

mechanisms the pheromone infrastructure, an extension to agent runtime environments, is introduced. The remainder of the section discusses how the deployment of a pheromone infrastructure eases the application of the proposed design principles.

After the informal introduction of the pheromone infrastructure in Section 3.1, the following Section 3.2 sets up a formal model of the infrastructure, proves the general global stability of the infrastructure, and demonstrates in a variety of scenarios how the model serves to predict, tune, and evaluate stigmergetic agent interactions.

The chapter ends with the presentation of an agent-based implementation of the pheromone infrastructure in Section 3.3. First, two additional extensions to the infrastructure are introduced that help to unload computations from the agents to local servers. Then, the agents comprising the implementation are specified in detail. Finally, requirements for an agent runtime environment that supports the pheromone infrastructure are stated.

In Chapter 4, the design, tuning, and evaluation of a self-organizing manufacturing control system is demonstrated. The guided manufacturing control (*GMC*) system for discrete flexible flow-shops is designed to combine robustness and flexibility with optimization for production goals. The design of the agents and their interactions follows the proposed design principles. The pheromone infrastructure is used in multi-agent coordination.

The guided manufacturing control system is discussed in the context of a car-body paint-shop application. Section 4.1 shortly presents the characteristics of the application. The designer is guided in distributing the agents in the manufacturing system and the system architecture is specified. In the Sections 4.2, 4.3, and 4.4 the agents and their interactions in the three bottommost layers are specified in detail and in the following Section 4.5 the emerging behavior of these layers is discussed and evaluated in a demonstration taken from the paint-shop application. The first layer realizes a reactive control of the manufacturing process, the second layer prepares the coupling of the reactive control with an advisory system, and the third layer translates global material flow goals into local advice.

Chapter 5 points the way to future extensions of the guided manufacturing control system, introducing concepts and deriving an agent model. Section 5.1 adds two more layers to the advisory system, providing strategy ranking, strategy evaluation, and strategy generation. The visualization system in Section 5.2 guides the system operator to critical locations in the production system.

The „synthesis“ (Chapter 6) visits the major issues of the thesis again, taking a more general perspective. In Section 6.1 the occurrence and relevance of the design principles proposed in Section 3.1 is systematically considered in the example of the previously presented manufacturing control system. Section 6.2 discusses the implications of the pheromone infrastructure and its formal model for the general evaluation of emergent global properties. Finally, Section 6.3 returns to the subject of manufacturing control in an attempt to classify different levels of sophistication in control by considering the manufacturing operation in state space.

The thesis concludes in Chapter 7 with a summary of the presented research and an outlook to further research directions.

# Chapter 2 - Background

*The following chapter provides the context for this thesis. The thesis has two major themes: (a) the general support of the design of synthetic ecosystems (Chapter 3), and (b) the exemplary development of a complex synthetic ecosystem for manufacturing control (Chapter 4). Accordingly, the first section in this chapter focuses on concepts and techniques related to synthetic ecosystems in general, while the second section provides a short review of the state-of-practice and the state-of-the art in manufacturing control.*

## 2.1 Concepts and Techniques

This thesis investigates systems made up of a number of separate entities, which interact among each other. As a result of these local interactions a global system-level behavior is observed. There are several branches in science where such systems are studied and each branch sets its own focus and uses its own techniques. Only very recently, interdisciplinary research has begun to emerge.

In the following, some of the basic concepts and technologies in the research into distributed systems are presented. The review presents approaches developed in biology, physics, chemistry, and in different branches of computer science. The purpose of the short review is not to cover all research activities and all the specific results gained. Rather, by looking at the same class of systems from different perspectives, a basic understanding of the characteristics of such systems should be conveyed.

### 2.1.1 Equilibrium Statistical Physics

In physics complex system-level behavior emerging from local interactions of spatially distributed individuals is studied in equilibrium statistical physics [Ashcroft and Mermin, 1976][Reif, 1965]. The entities in the system are simple physical objects – often called particles. There is no centralized control of any sorts. Local deterministic laws that are superimposed with probabilistic noise processes specify the interactions among the particles. The application of techniques from equilibrium statistical physics usually requires a huge number of particles (e.g., $10^{23}$).

The analysis of such particle systems focuses on the stable state character of the system. It is assumed that with ongoing interactions on the particle level, the overall system behavior settles in a stable state. The location of the stable state is dictated by the specific characteristics of the local interactions. Equilibrium statistical physics has developed a number of mathematical techniques. These are, for instance, the mean field theory, self-averaging approximations, analysis of phase transitions, Monte Carlo techniques, the replica trick, or tools to analyze the thermodynamic limit when the number of particles approaches infinity [Wolpert and Tumer, 1999].

An open issue in research is the adaptation of the model to a system of more complex individuals so that such techniques may also be applied to analyze their emerging system-level behavior. There are already examples in literature pointing the way to solutions. The iterated prisoner's dilemma played on a lattice is investigated numerically [Szabo and Toke, 1998], stochastic games are analyzed by expressing the deviation from rationality in

the form of a „heat bath“ [Marsili and Zhang, 1998], the complexity of a voting system is quantified on the basis of its topological entropy [Meyer and Brown, 1998], and the required simplicity of individuals that produce sufficiently complex system-level behavior is investigated [Delgado and Sole, 1997].

### 2.1.2 Cellular Automata

Another widely used technique to analyze particle systems is to represent the system and the underlying physical laws in a cellular automata model. Ulam and Von Neuman have already conceived cellular automata in the 1940s. The model provides a formal framework for an investigation of the behavior of complex, extended systems [von Neumann, 1966]. In the 1980s, Wolfram added substantially to the body of knowledge.

A cellular automata model is a regular n-dimensional grid of cells where each cell is a finite state machine. Usually, cellular automata models are considered in one, two, or three dimensions, permitting finite or infinite numbers of cells. The state of the system changes in time by changing the state of each automaton according to its local interaction rules. In most models time is discrete, the update occurs synchronously, and all automata share the same rule.

The interaction rule, also known as the transition function, determines the state of an automaton on the basis of its own previous state and the state of the cells in a surrounding neighborhood. A radius specifies the neighborhood of a cell. The rule maps every possible configuration of states in the neighborhood to a new state for the respective cell. Often, a rule table is used to specify the transition function.

Cellular automata models are applicable to a wide range of research topics. They are used in the study of general phenomenological aspects of the world, including communication, computation, construction, growth, reproduction, competition, and evolution [Margolus and Toffoli, 1987] [Burks, 1970] [Smith, 1969] [Perrier et al., 1996]. Since the late 1960s, Conway's „game of life“ [Gardener, 1970] [Gardener, 1971] fascinates researchers in Artificial Life (Section 2.1.4). The „game“ specifies one of the most well known rule sets for cellular automata. Later, the set of „game of life“ rules was shown to be computation-universal [Berlekamp et al., 1982]. In physics cellular automata provide discrete models for a branch of dynamical systems theory that studies the emergence of well-characterized collective phenomena such as ordering, turbulence, chaos, symmetry-breaking, fractality, etc. [Vichniac, 1984] [Bennett and Grinstein, 1985] [Wolfram, 1984] [Wolfram, 1983]. There, research departs from the equilibrium statistical physics (Section 2.1.1) and investigates dynamics outside static stability points. Finally, cellular automata models find their way into biological modeling as a review in [Ermentrout and Edelstein-Keshet, 1993] shows.

### 2.1.3 Population Biology and Ecological Modeling

Similar to equilibrium statistical physics (Section 2.1.1), population biology and ecological modeling is concerned with the large-scale „emergent“ processes when a large number of (relatively) simple entities interact with one another [Begon et al., 1996] [Hastings, 1997]. But, while in physics these entities are particles, in the related biological field members of one or more species are the individuals in the model. Hence, in practically relevant systems the number of entities is much smaller than in physics.

The interactions among the separate entities in the system are an abstraction of the process of natural selection as it occurs in biological systems. Usually, processes like genetic reproduction, genotype-phenotype mapping, or inter and intra-species competition for resources are modeled. As in its counterpart in physics, population biology and ecological modeling focuses on the dynamics of the resulting ecosystem and its long-term behavior depending on the local interactions.

The results gained in population biology and ecological modeling are applicable to non-biological systems too. For instance, social issues like the emergence of language or culture, warfare, and economic competition are investigated [Epstein and Axtell, 1996] [Gabora, 1998]. Population models lend themselves to the research into the behavior of large complex systems with many interacting components [Hanski et al., 1996] [McFarland, 1994] [Nishimura and Ikegami, 1997] [Polls, 1998].

### 2.1.4 Artificial Life

Research into artificial life links biology with the field of computer science. But it also draws from physics, chemistry, economics, and philosophy. Langton defines artificial life research as follows:

*„A field of study devoted to understanding life by attempting to abstract the fundamental dynamical principles underlying biological phenomena, and recreating these dynamics in other physical media, such as computers, making them accessible to new kinds of experimental manipulation and testing.“*

—[Langton, 1992]—

Artificial life research has two major objectives. First, it seeks to understand the abstract functioning and especially the origin of terrestrial life. And second, it attempts to create artificial organisms that can meaningfully be called “alive”.

Formalizing and abstracting the mechanical processes underpinning terrestrial life serves the first objective. One fundamental process investigated here is the assembling of lipids into more complex structures such as vesicles and membranes [Deamer and Oro, 1980] [Edwards and Peng, 1998] [New and Pohorille, 1999]. On a more abstract level, the processes of self-replication [Breyer et al., 1998] [Smith, 1992] [von Neumann, 1966] and of functional self-organization [McMullin and Varela, 1999] are examined.

The second objective of artificial life research is less analytical and more creative. It takes terrestrial life as an inspiration and designs living systems. The design of an immune system for computers remains very close to the biological model. Such a system develops “antibodies” that fight computer viruses, which many consider a life-form in its own right, more rapidly and more efficiently than other algorithms [Forrest et al., 1994] [Kephart, 1994] [Hofmeyr and Forrest, 1999]. The development of software through evolution [Koza, 1992] is a more abstract application of biological principles in systems design. Finally, the creation of artificial worlds inside a computer to study general forms of life (“Life-As-It-Could-Be”) serves both objectives since, up to now, life on Earth is the only real-world sampling point available to understand life.

Decentralized systems with many interacting individuals abound throughout the terrestrial biosphere and artificial life research investigates such systems too. One famous example of a distributed artificial life system is the model of flocking behavior [Reynolds, 1987] as it is encountered in birds. The individual entity in the distributed system is called

“Boid”. It represents a virtual bird with basic flight capabilities. The flock, the global system, is a collection of Boids in a simulated world.

Each Boid has a restricted perception of its local environment. It can only perceive nearby flock-mates or obstacles. Global perception, as provided by maps for instance, is not permitted. The behavior of each Boid as it moves through the virtual space is expressed by three simple rules:

**Collision Avoidance (1)**.—Avoid collisions with nearby flock-mates or obstacles.

**Velocity Matching (2)**.—Attempt to match velocity with nearby flock-mates.

**Flock Centering (3)**.—Attempt to stay close to nearby flock-mates.

As an effect of the local activities of each Boid, the flock moves in a cohesive group that spontaneously splits into subgroups when encountering obstacles. After the obstacle is cleared, the subgroups join again. The observation of Boids has an analytical and a practical implication. Analytically, the fact that tuning the parameters of the behavior results in flight patterns analogue to different bird species indicates that similar rules dictate collective animal behavior too. As a practical consequence, photo-realistic imagery of swarms has been produced in computer animation. These animations have been used to model bat swarms for several movies already.

The flocking behavior demonstrated in the Boids gives rise to a new search heuristic in optimization. Particle swarm optimization [Kennedy and Eberhart, 1995] [Ozcan and Mohan, 1999] replaces the 3-dimensional computerized world the Boids “live” in with the n-dimensional (continuous) space of solutions to an optimization problem. Thus, the current location of a Boid represents one solution. Similar to genetic search methods, a flock of Boids in a search space is a set of solutions currently examined.

The behavior of each Boid is extended by a fourth rule that probabilistically biases the individual towards better solutions (local gradient in search space). The cohesion of the flock, provided by the other three rules, ensures that local search (hill climbing) is tempered by more global information coming from nearby individuals. The probabilistic component in the local search guarantees that the whole search space is covered eventually by random walk. While the heuristic in traditional genetic search (outcome of genetic recombination) may select two good solutions from different ends of the search space, particle swarm optimization produces new solutions (next Boid position) dynamically guided by a number of neighbors [Eberhart and Shi, 1998].

Particle swarm optimization generally requires a continuous search space. The effectiveness of the heuristic depends on the settings of the parameters that specify the four individual behaviors. These parameters effectively select the “bird species” modeled in Reynold's original flock of Boids. As each bird species has adapted its flocking behavior to its specific ecological niche, the particle swarm must be adapted to the respective search space. In [Shi and Eberhart, 1998] first steps towards parameter adaptation are taken.

### 2.1.5 Multi-Agent Systems

Historically, the behavior of the individual entity is often rather simple in systems explored in physics, biology, and even in artificial life. Only recently, more complex individual behavior is considered. The “trajectory” of research runs from simple to complex individual behavior.

On the other hand, artificial intelligence (AI) initially was primarily concerned with singular entities like expert systems. Research into distributed artificial intelligence (DAI) only set in when traditional AI tasks were implemented in parallel. Such implementations either parallelized the AI production system or the underlying programming language [Forgy, 1982] [Rich and Knight, 1991], or different modules with different tasks (e.g., reasoning, planning, scheduling) concurrently attempted to achieve a common goal [Huhns, 1987] [Iyer and Ghosh, 1995] [Lee et al., 1999].

The concern of DAI, especially of its sub-field "Distributed Problem Solving" (DPS), is how to modularize a given task efficiently. DPS is defined as follows:

*"Distributed Problem Solving considers how the work of solving a particular problem can be divided among the number of modules that cooperate at the level of dividing and sharing knowledge about the problem and developing a solution."*

—[Bond and Gasser, 1988]—

DPS solutions often involve a centralized controller that allocates tasks to the different modules and then processes the associated results. Since such architectures are often brittle, more recently there has been a move towards more autonomous modules and fewer restrictions in the interactions among the modules. But DAI still maintains the traditional AI concern, focusing on particular aspects of intelligence, such as reasoning, understanding, learning, etc.

The field of Multi-Agent Systems (MAS) research has its roots mainly in DAI. It is based on the idea that intelligence should emerge from the interactions among components without constructing it explicitly into them [Minsky, 1986] [Brooks, 1991b] [Brooks, 1991a] [Jennings et al., 1998]. Hence, multi-agent systems are similar to the distributed systems observed in physics or biology in the sense that multiple individuals (here: agents) interact and complex system-level behavior is observed. But, agents are traditionally much more complex than, for instance, particles or Boids. There has been a long discussion how to define an agent that never really came to a conclusion. A thorough review of the different definitions is presented in [Franklin and Graesser, 1997].

As the single agent may be a complex system in itself, MAS research has two major foci: the inner workings of each agent, and the interactions among agents [Bradshaw, 1997], [Sycara, 1998] [Jennings et al., 1998]. But, the final objective is to organize the agents to achieve some global task. Thus, several techniques are developed (e.g., coordination [Sen et al., 1994], negotiation [Kraus, 1997], coalition forming [Sandholm et al., 1998] [Sandholm and Lesser, 1995] [Zlotkin and Rosenschein, 1994], contracting [Andersson and Sandholm, 1998].

**Design Methodology**

MAS research is a field in computer science and hence it has a strong engineering aspect. Whereas equilibrium state physics, for instance, takes a system of interacting individuals and asks what are the emerging features of the system, MAS researchers are concerned with the inverse problem: given a specified global behavior, what individual behavior is required to achieve it. To solve the inverse problem, design methodologies for multi-agent systems are developed.

One example of a multi-agent design approach is presented in [Burmeister, 1996]. The approach is derived from the successful object-oriented system design, which dominates

current software engineering. The agent-oriented design process results in three sub-models specifying the system: the agent model, the organizational model, and the cooperation model.

The agent model contains agents and their internal structure. To derive the agent model, first the agents and their environment are specified. Then for each agent its motivations, its behavior, and its knowledge are laid out.

The organizational model specifies the relationships among agents and agent types. These may be inheritance relations (among agents and agent types, and agent types and sub- or super-types), or relationships among agents based on their roles in organizations. These organizations may be means for structuring a complex system into subsystems (as done in some object-oriented techniques) or they may be used to model real organizations. The organizational model is constructed by identifying the roles in the respective scenarios, by building inheritance hierarchies, and, finally, by structuring roles into organizations.

The cooperation model describes the interaction or more specifically the cooperation among agents. It contains the "dynamics in the large" only. The "dynamics in the small" (i.e., the description of agent behavior) are part of the agent model. To design the cooperation model the cooperations and the cooperation partners are identified first. In a second step, the message types and the cooperation protocols are specified.

### 2.1.6 The Artificial Life Road to Artificial Intelligence

At the intersection of traditional multi-agent systems research, biology, and artificial life the so-called "Artificial Life Road to Artificial Intelligence" [Steels, 1995] opens up. Here, researchers seek to apply concepts inspired by biological systems to the design of (intelligent) multi-agent systems. A similar approach is found in a related area of AI that researches artificial neural networks.

Very close to the biological model are computational ecologies [Huberman and Hogg, 1988]. Computational ecologies are related to the field of ecological modeling (Section 2.1.3). They are large distributed systems with independently acting individuals. But, the mathematics of the interaction model need not be derived directly from biological processes. Implementations of computational ecologies are often based on cellular automata instead of independently acting software entities.

Another approach that is more related to Reynold's model of flocking behavior (Section 2.1.4) explores active walker models. Physicists have initially pioneered these models [Batty, 1997] [Helbing et al., 1997a] [Helbing et al., 1997b]. Active walkers, which may be humans or simple physical objects, cross a field along trajectories. While they cross the field the walkers leave trails behind. The trajectory chosen by a walker optimizes its private utility function, which is influenced by several factors including in particular existing trails. The specific concern in active walker models is how to design the field (e.g., introducing obstacles or paths), so that some required trail patterns emerge, assuming that the internal utility function cannot be changed. A practical application of active walker models is the design of cement pathways that are actually followed by human walkers.

The "Physics-Agent-System Model" [Shehory et al., 1999] is another adaptation of a physical system to model the interaction and coordination in large fine-grained multi-agent systems. Here agents and goals that require agent coordination are modeled as particles in

a (simulated) world. Each particle has its own mass and location and the particles that represent an agent also move with a direction and speed that changes over time. The mass of a goal particle relates to the importance of the goal whereas the mass of an agent particle matches the portion this agent may bring into the goal fulfillment. The movement of the agents results from gravitational forces among the particles and the coordinated goal fulfillment is triggered by the collision of agent and goal particles.

### Swarm Intelligence

In recent years swarm intelligence has gained increasing popularity. Swarm intelligence is broadly defined as follows:

*"Any attempt to design algorithms or distributed problem-solving devices inspired by the collective behavior of social insect colonies and other animal societies."*

—[Bonabeau et al., 1999]—

The approach is motivated by the realization that the rich behavior of social insect colonies arises not from the sophistication of any individual entity, but from the interaction among these entities. It has even been proposed that the more complex a society, the more simple the individual may be [Delgado and Sole, 1997].

Historically, early attempts to incorporate social animal mechanisms into systems design can be found in the work of Tselin and Rabin. Rabin [Cook and Racko, 1980] [Rabin, 1982] constructed moving automata that solve problems on graphs and lattices by interacting with the consequences of their previous actions. Tsetlin [Tsetlin, 1973] suggested that especially randomness, decentralization, indirect interactions, and self-organization are characteristics that make the swarm-based approach potentially powerful.

In general, the objective of swarm intelligence is to uncover the kinds of interactions among the entities that lead to some pre-specified system-level behavior. Furthermore, learning in large distributed systems is investigated.

#### *Collective Intelligence*

Collective Intelligence (COIN) [Wolpert and Tumer, 1999] can be seen as one instance of swarm intelligence that specifically focuses on the learning issue. Wolpert and Tumer define the following characteristics for a COIN: Many processors (agents) run concurrently, performing actions, which affect the behavior of other processors. Centralized and personalized communication or control is not permitted. Finally, there has to be a well-specified task set for the entire distributed system that typically requires extremizing some utility function.

To achieve system features like scalability, wide applicability, little hand tailoring, robustness, or adaptivity the single processors in COIN models run a local reinforcement learning mechanism. The local reinforcement learning mechanisms are fed by external "reward" and "penalty" signals. The learning approach does not require a "teacher". It may be model-free and it operates on-line. On the basis of the COIN model a formal specification of the optimal agent design and initialization for a given problem is attempted.

### *Stigmergy*

A fundamental principle in the emergence of coordinated system-level behavior from the local interactions of individuals is stigmergy. The French biologist Grassè already introduced the concept in the 1950s [Grassè, 1959] [Grassè, 1984] but only now it becomes a major issue in agent system design. Grassè discovered the principle when studying the behavior of social insects. The name stigmergy is a combination of the Greek words "stigma" (outstanding sign) and "ergon" (work), indicating that some activities of agents are triggered by external signs, which themselves may be generated by agent activity. Thus, simple activities may be coordinated by indirect communication and robust phenomena may emerge that remain virtually unchanged even under heavily changing circumstances.

Two general forms of stigmergy are identified. Sematectonic stigmergy involves a change in the physical characteristics of the environment. Termite nest building is an example of sematectonic stigmergy in multi-agent coordination. An individual observes a developing structure and adds to it. The second form is sign-based stigmergy. Here, some marker is deposited in the environment that makes no direct contribution to the task being fulfilled but influences subsequent task related behavior.

Stigmergy does not explain the detailed coordination mechanisms. But, for the inverse problem of designing a system to fulfill a global task, stigmergy provides a general concept that links individual and colony-level behavior. The advantages gained by applying stigmergy are simple agents, reduced communications, the incremental construction and improvement of solutions, and the flexibility of the system-level behavior in the face of disturbances.

Social ants exhibit both forms of stigmergy. Sematectonic stigmergy is observed, for instance, in the piling of dead ants. Ants fulfilling the piling task show the following probabilistic behavior:

**Rule (1)**.—Move randomly over the field.

**Rule (2)**.—If you find a dead ant at point x in the field and if you do not carry one already then pick it up with a probability $P_{up}(f(x))$.

**Rule (3)**.—If you carry a dead ant then drop it with a probability $P_{down}(f(x))$.

The probabilities $P_{up}$ and $P_{down}$ depend on the density f(x) of dead ants in the vicinity of location x. An ant perceives the density by keeping a short-term memory of the number of dead ants encountered in its random walk. The higher the density of dead ants, the lower is $P_{up}$ and the higher is $P_{down}$.

An example of sign-based stigmergy is the behavior of ants when they collectively construct the shortest path from a food source to the nest [Beckers et al., 1992] [Goss et al., 1989]. It has been shown that in general the ants do not use any visual input in their activities [Hölldobler and Wilson, 1990] and, there is also no centralized reasoning. But still, the ants find their way even if introducing obstacles changes the landscape. The underlying mechanism of the adaptive optimization is shown in the following scenario. The figures used to illustrate the example are taken from [Dorigo, URL].


Figure 2.1. Straight Pheromone Trail

Figure 2.1 displays the initial scenario. The ants move on a straight path that is marked with pheromones. The path connects the food source with the nest. A pheromone is a specific chemical substance that is generated and perceived by the ants. It is the medium of indirect communication in the scenario. The ants on their way from the food source to the nest deposit a certain amount of pheromone while they walk and they probabilistically prefer to move into a direction where the pheromone trail is strongest.

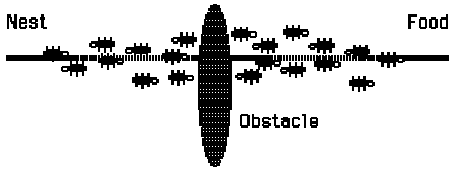


Figure 2.2. Obstacle Introduced

When the existing trail is obstructed (Figure 2.2) the ants arriving at the obstacle face a random choice of going left or right. Without any pheromone information available the flow of ants splits roughly in half, going around the obstacle. The random choice explores both options (Figure 2.3).


Figure 2.3. Two Options are Explored

On the shorter path the ants arrive faster back at the original trail permitting a faster flow between the nest and the food source. Statistically, a higher rate of pheromone deposits on the shorter path than on the longer one results. Thus, taking evaporation of pheromones into account, the shorter path reaches higher pheromone strength, attracting more ants until all ants are recruited to the short path (Figure 2.4).

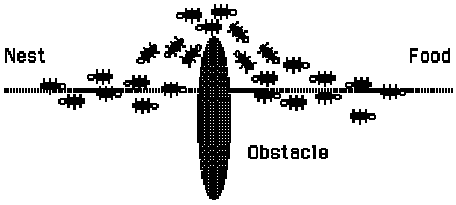


Figure 2.4. Shortest Path Dominates

The food foraging behavior of the ants is discussed in more detail in Section 3.1.4 where it serves as an example for naturally occurring self-organization.

### *Ant Colony Optimization*

Ant colony optimization is a specific application of the swarm intelligence approach that seeks to adapt coordination mechanisms employed in social ant colonies to solve discrete optimization problems. The ant colony optimization meta-heuristic [Dorigo and Di Caro, 1999] [Dorigo et al., 1999] has its roots in the ant system presented originally in the Ph.D. thesis of Dorigo in 1992.

Ant systems are applied to the traveling salesman problem [Dorigo et al., 1996]

[Colorni et al., 1993] [Dorigo and Gambardella, 1997] [Bullnheimer et al., 1999] and to the quadratic assignment problem [Maniezzo et al., 1994] [Taillard and Gambardella, 1997] [Maniezzo, 1998] [Gambardella et al., 1999] [Maniezzo and Colorni, 1999] [Stützle and Dorigo, 1999]. Recent extensions hybridize ant systems with Q-learning (Ant-Q) and incorporate local search.

Ant colony optimization is also successfully applied to dynamic real-world problems like routing and load balancing in circuit switched telecommunications networks [Schoonderwoerd et al., 1996] [Schoonderwoerd et al., 1997] and routing in packet switched telecommunications networks [Di Caro and Dorigo, 1998a] [Di Caro and Dorigo, 1998b] [Varela and Sinclair, 1999]. An application to connection management and diagnostics in communications networks is presented in [White and Pagurek, 1998] that explicitly operates on a generic „chemistry“ of pheromones.

The processing of local information in a shared physical environment and its incorporation into probabilistic decision making as demonstrated by the ants' pheromone-based coordination mechanisms also inspired a reactive chess-playing algorithm [Drogoul, 1993]. In the MARCH system individual game figure strength and material value is propagated to the threatened fields before a move. There these values combine locally to create a weighting of the different possible moves. The actual move is then selected probabilistically based on these weights. Even though this architecture does not make any use of the ants' swarming behavior it nevertheless achieves an acceptable playing performance.

***Synthetic Ecosystems***

According to the definition of swarm intelligence, the synthetic ecosystems approach [Parunak, 1997] [Parunak et al., 1998] applies swarm intelligence to the design of multi-agent systems. The main concern of research into synthetic ecosystems is to provide practical engineering guidelines to design systems of industrial strength.

In its practical application the synthetic ecosystems approach is not bound to apply the actual social animal coordination mechanisms to the problem at hand. Rather, the proposed design guidelines seek to capture the underlying logic of the biological and other complex systems. Examples for design guidelines are: Identify agents according to things, not functions, keep agents small, decentralize system control, support risky redundancy, enable agents to share information, and plan and execute concurrently.

These are all general design guidelines that are orthogonal to design methodologies as they are proposed in multi-agent systems research (Section 2.1.5). As a consequence, the three models proposed in [Burmeister, 1996], for instance, may still be constructed in the design phase. The design principles from synthetic ecosystems research apply to the intermediate steps that lead to the models.

### 2.1.7 General Concepts

Two concepts are often invoked in discussions of global features of distributed systems when appealing to the intuition of the reader. Many interesting features are said to be “emergent” or “self-organized”.

### Emergence

There is still an ongoing discussion in the scientific community, if emergence really exists or if it is only a trick of perception. The discussion is not restricted to computer science. Rather, it has been a philosophical question for at least one and a half centuries [Mill, 1843]. One of the more successful attempts to define the concept is to be found in "The Dictionary of Philosophy of Mind":

*"Properties of a complex physical system are emergent just in case they are neither (i) properties had by any parts of the system taken in isolation nor (ii) resultant of a mere summation of properties of parts of the system."*

—[Eliasmith, URL]—

This definition is a rather general, since it does not explain where the emergent properties of the systems may come from. In the exploration of distributed systems of locally interacting components the interactions are seen as the source of emergent properties. But it is also important that the local individuals do not perceive the global property. There are different levels of perception in the system. One set of properties is perceived and handled at the level of the individuals (e.g., velocity of a particle), while another set of properties dominates the system level (e.g., temperature of a particle system).

For the purpose of this thesis, the separation into levels of perception should be sufficient to capture the essence of emergence. In the design of a distributed system that fulfills some specified requirements, the designer has to know how a specific system-level property is influenced by local knowledge and interaction patterns. In the specification of the system it must be clear what the current context is. There are the definitions of the local behavior of the individuals and there are system-level specifications. The local part may not incorporate global properties and vice versa. Only when arguing the translation from local to global features (emergence) both property sets are discussed.

### Self-Organization

The concept of self-organization originally emerged in the context of physics and chemistry [Haken, 1983] [Nicolis and Prigogine, 1977] to describe the emergence of macroscopic patterns out of processes and interactions defined at the microscopic level. More recently, self-organization is also invoked in reference to colonies of social insects where wide ranges of collective phenomena are self-organized [Deneubourg et al., 1989]. The advantage of interpreting complex colony-level behavior on the basis of self-organization is that it does not require the invocation of complex behavior at the individual level.

Self-organization is seen as a set of dynamical mechanisms in a system where structures appear at system levels that are not externally imposed. As it is already discussed in the context of emergence, self-organization requires that the local interactions of the individuals do not reference to these global structures.

For a system to show self-organized properties, the following ingredients are proposed [Nicolis and Prigogine, 1987]. It is still subject of research to formally validate the items in the list.

- The system has nonlinear dynamics. There is positive (e.g., recruitment, reinforcement) and negative (e.g., saturation, exhaustion, competition) feedback.
- The internal mechanisms include randomness and amplify random fluctuations that break symmetries (e.g. random walks, errors, random task-switching).
- Self-organized properties are generated by multiple, local interactions in space and time (e.g., stigmergy) that usually require a minimal density of individuals.
- The system is thermodynamically open to prevent a thermodynamic equilibrium ("heat death").
- There is a continuous change where dissipative structures "export entropy" to environment.

Besides these (assumed) requirements for self-organized properties in a system, the following indicators of self-organization should be followed [Bonabeau et al., 1999]:

- Spatio-temporal structures arise in an initially homogeneous medium.
- There exist several stable states in the system-level behavior (multi-stability) and the one that is actually reached depends on the initial conditions.
- Slight variations of some system parameters may lead to dramatic changes in system behavior (bifurcations).

### 2.1.8 Summary

Synthetic ecosystems research is the application of swarm intelligence to the design of multi-agent systems. The primary focus lies on the problem of designing individual agent behavior to achieve a pre-specified system level behavior. This thesis provides designers of such systems with a set of guidelines supporting a systematic engineering of stable self-organized behavior. These guidelines seek to capture the underlying (bio-)logic of interaction mechanisms in large-scale natural agent systems like social insect colonies and integrate it into the software engineering process.

Besides supporting the design of swarm-intelligent systems for industrial applications, the formal evaluation of the global behavior of these systems is a major issue in research. This thesis sets up a foundation for the formal description of stigmergetic agent interactions and their analysis. On the basis of such a formal model emerging patterns of system-level behavior may be predicted and the fulfillment of the global requirements may be evaluated. Starting out with formal models of multi-agent behavior, future research may be able to integrate methodologies adapted from statistical physics or population biology to further improve the evaluation process.

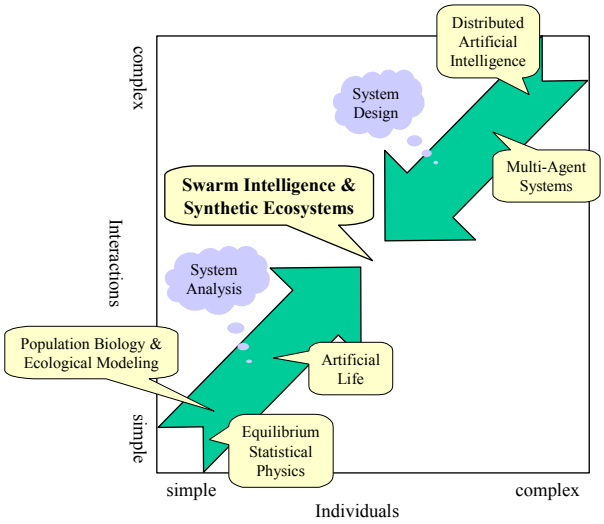


Figure 2.5. “Trajectories” of Research into Distributed Systems

The integrative approach followed in this thesis is illustrated in Figure 2.5. Traditional research into multi-agent systems has focused mainly on the systems design aspect, while branches in natural sciences like population biology seek to analyze emerging global behavior. Industrial strength system engineering eventually requires both, design and analysis of distributed systems of interacting individuals.

## 2.2 Manufacturing Control

The design and formal evaluation of synthetic ecosystems is exemplified in this thesis on the basis of a manufacturing control application. The following review presents manufacturing control as part of the production processes and characterizes manufacturing systems as they are encountered today. In a second step, requirements on tomorrow's control systems are derived from the current change in the environment businesses operate in. As a consequence of these changes, distributed manufacturing control systems that self-organize dynamically in a bottom-up fashion have to be developed. The last step of the review classifies systems following such an approach as they are proposed in research.

### 2.2.1 Manufacturing Control in Context

#### Production Processes

Activities in production are differentiated into the physical production and manipulation of products on the one hand, and the management and control of these physical activities on the other hand [Bongaerts, 1998]. According to its perspective, management and control is further divided into strategic (long-term), tactical (mid-term), and operational (short-term) activities.

Strategic production management determines the set of products of the company. It considers the markets and the expectations of potential customers. Long-term management also designs the manufacturing system for the respective products, including a master schedule.

Tactical production management takes the master schedule and generates detailed production plans, which comprise release and due dates for assemblies, subassemblies and components. In make-to-order production, mid-term management operates on real customer orders, while make-to-stock production uses predicted orders. Today's planning processes are usually supported by MRP systems (Material Requirements Planning) or MRP-II systems (Manufacturing Resource Planning).

Operational production management handles the manufacturing processes on the shop floor. The schedule generated by the mid-term planning process is translated into detailed plans that specify the appropriate resource for every task and the optimal sequence of all tasks on every resource. Short-term management explicitly coordinates the machines, workers, tools, and other resources, usually considering a period spanning a day up to a few weeks.

Short-term production management does not take the current situation into account when it allocates the resources. In most of today's production systems it is the task of the manufacturing control system to implement the detailed short-term schedule generated by the operational production management. During the implementation immediate reactions to disturbances are required, which often invalidate the schedule.

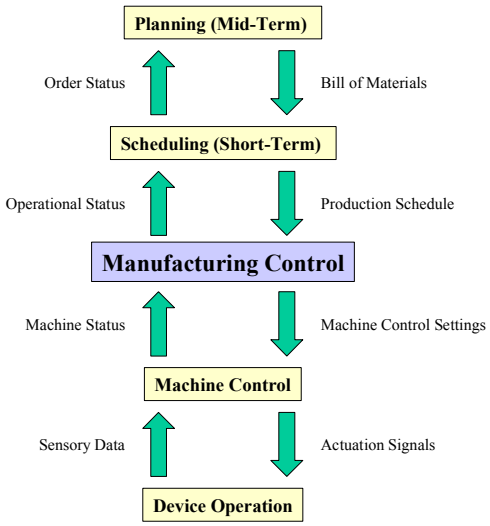


Figure 2.6. Production Management and Control

Figure 2.6 illustrates the position of the manufacturing control process in the traditional hierarchy of production management and control. In literature, manufacturing control systems are also found under the name “manufacturing execution systems” (MES).

Most of today's automated systems that support the production management process mirror the hierarchical structure of the processes. The hierarchical system architecture reduces the complexity of the development and makes them easier to change compared to monolithical systems.

Decisions in the planning phase are taken in a top down fashion. In mid-term planning real or assumed customer orders are given release dates and due dates, which in turn determine the resource allocation specified in short-term scheduling. Finally, the optimized schedule is handed over to the manufacturing control system for implementation. In case of disturbances during schedule execution the planning process may have to restart at some point because the schedule became infeasible. The hierarchical approach is rigid, modifications are costly, and intelligent reaction to disturbances during execution is not

considered.

**Characterizing Manufacturing Systems**

There are several dimensions by which manufacturing systems may be characterized. The type of the material flow distinguishes continuous and discrete manufacturing systems. Mass production, large batch manufacturing, and one-of-a-kind manufacturing are identified when the volume of production is considered.

The layout is a third dimension that differentiates manufacturing systems. In discrete manufacturing single machine models, identical processors models, flow shops, job shops, and recently, flexible flow shops are encountered. In single machine models there is only one machine fulfilling all operations. Identical processors models assume multiple but identical machines, whereas in flow shops different machines are permitted. But then, flow shop production requires each operation to be executed on a specific machine and the sequence of all operations of an order is fixed. Additionally, the sequence of orders visiting machines is fixed too. The latter requirement does not hold for job shops. In job shops orders might overtake each other during execution.

Flexible flow shops are presumed to retain the efficiency and transparency of a flow line, combining it with the reactivity and flexibility of a job shop. Flexible flow shops increase flexibility in machines selection, operation sequencing, and routing in high-volume production. An important characteristic of the control of flexible flow shops is that of all feasible routes through the system only a small subset is preferable in terms of the production goals. The composition of the subset changes dynamically when disturbances occur and it is not always obvious what routing to choose in a given situation.

### 2.2.2 Requirements of Tomorrow's Manufacturing Control

The subsequent argument follows a discussion in [Bussmann and McFarlane, 1999]. It is illustrated in Figure 2.7, which links (marked squares) the changing business environment with the requirements of modern manufacturing control systems.

Driven by globalization and the growing surplus of industrial capacity, more and more businesses are faced with a change from a supplier's market to a customer's market. The trend empowers the customers to demand, for instance, constant product innovation, low-cost customization, and better service. As a consequence, companies have to shorten product life cycles, reduce time-to-market, and increase product variety. At the same time the quality of the products has to be maintained and the investment costs must be reduced to remain competitive.

Future manufacturing systems are faced with an increasing complexity in processes and products, they have to cope with continual change, and the costs in manufacturing have to go down. In detail, products will have more features and more variants combined with a shorter time-to-market, a decreased life cycle, and lower investment per product. Additionally, the volume of a product manufactured and the mix of variants requested from the manufacturing system varies strongly over time.

Such a drastic change in the external conditions of production must be answered by changes in the manufacturing system. To handle the process complexity, manufacturing systems must be restructured in an intuitive and self-explaining way and they must

guarantee a well-defined behavior in a wide range of situations. Furthermore, flexibility and reconfigurability of manufacturing units and processes must increase to support re-use and scalability of the system. Finally, the manufacturing process must be robust.

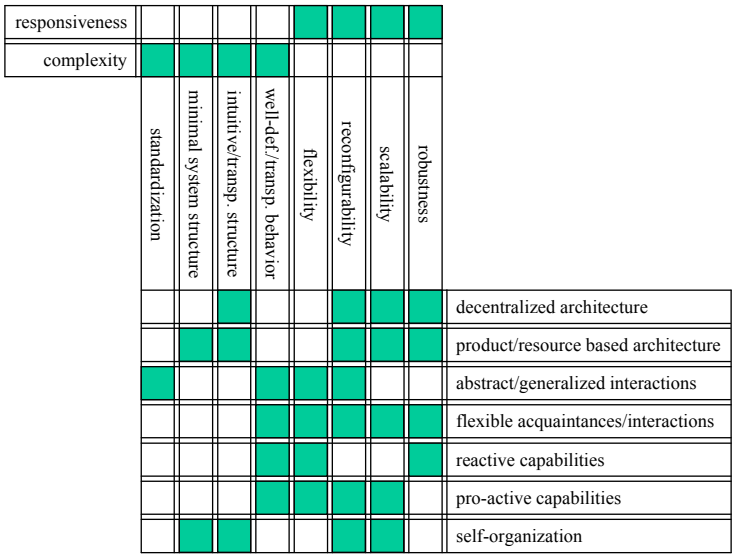


Figure 2.7. Linking Business Trends with Control Requirements

A new type of manufacturing system also requires a new approach to manufacturing control. The architecture of the control system must be decentralized on the basis of a product, order, and resource partition. The resulting interactions in the decentralized system should be abstract, generalized, and flexible. The control system as a whole has to take reactive and proactive actions concurrently in a self-organized manner.

### 2.2.3 Distributed Architectures for Manufacturing Control

#### Holonic Systems

Distributed systems of interacting individuals for production management and control are considered by the academic community and also by industry. Initially, research into distributed architectures for the manufacturing domain developed separately in production research and in the multi-agent systems community (Section 2.1.5). Only recently, these two strands recently grew together.

Following the specific requirements of modern manufacturing systems (Section 2.2.2), Suda [Suda, 1989] proposed a “holonic” model for designing and operating elements comprising manufacturing processes. The philosopher Arthur Koestler [Koestler, 1967] coined the term “holon” based on the observation that most entities encountered in nature are also part of other (higher-level) entities. Hence, a “holon” is a whole individual (Greek: “holos”) and a part (Greek suffix “on”) at the same time.

A holon in the manufacturing domain is defined as “an autonomous and cooperative building block of a manufacturing system for transforming, transporting, storing and/or validating information and physical objects” [Christensen, 1994]. A manufacturing holon comprises a control part and an optional physical processing part. Multiple holons may dynamically aggregate into a single (higher-level) holon.

A distributed system of interacting holons is called a “holarchy”. The holons in the system may cooperate to achieve a goal or objective. Holarchies are created and dissolved dynamically. The application of the holonic concept to the manufacturing domain is expected to yield systems of autonomous, cooperating entities that self-organize to achieve the current production goals. Such systems meet the requirements of tomorrow's manufacturing control systems.

Bussman argues in [Bussmann, 1998] that MAS research may provide the technology to realize the information processing part of a holon. In that sense, holarchies are multi-agent systems with specific structures and interactions; while holons are agents in the manufacturing context that may have physical abilities and that may aggregate. This thesis focuses on information processing and hence the term “agent” is used predominantly even in a manufacturing context.

### Dimensions of Control Architectures

Agent-based application architectures may be characterized in the following three dimensions:

**Individuals and Types**.—What becomes an agent?

**Interaction Structure**.—Who talks to whom?

**Interaction Contents**.—What is said and how is it said?

These dimensions are not independent. A design decision in the identification of individuals, for instance, influences the available interaction structures and the type of contents exchanged in the interactions, and vice versa. In accordance to these three dimensions, a classification of agent architectures for manufacturing control is presented.

#### *Individuals and Types*

When identifying the individuals in an application, the partition into agents can be entity-oriented or function-oriented. Most manufacturing control architectures proposed recently primarily follow the entity-oriented approach. Often, the resulting architectures in discrete manufacturing applications may be interpreted in terms of the PROSA reference architecture. This is the case, for instance, in the WEST architecture [Bussmann and Schild, 2000].

PROSA is a holonic reference architecture for manufacturing systems developed at K.U.Leuven [Van Brussel et al., 1998] [Wyns, 1999]. As reference architecture it specifies what kinds of holons should comprise the system, what the general responsibilities of these holons are, and what the structure of the interactions in the holarchy is. PROSA may serve as a starting point for designing specific manufacturing applications.

PROSA identifies three kinds of basic holons: Product-holons, Resource-holons, and Order-holons. A Product-holon encapsulates the process and product knowledge required for a correct manufacturing of the product with sufficient quality. It provides information to the other holons (e.g., product life-cycle, user requirements, design, process plan) without being able to take any physical actions itself.

The physical part of a Resource-holon is a production resource, and the control of the resource comprises the information processing part. Resource-holons provide production

capacity and functionality to other holons. They have the complete knowledge and abilities to organize, use, and control their production resource.

An Order-holon represents a manufacturing order. It has to make sure that all required tasks are performed correctly and in time.

The operation of the basic holons may be assisted by Staff-holons. These holons are intended to offer functionality that is typically found in higher levels of hierarchical control systems. But in difference to the hierarchical approach, Staff-holons only have an advisory role. The actual decisions are taken by the basic holons. The rationale behind the introduction of Staff-holons is to decouple the robustness and agility provided by the basic holons from optimization functions fulfilled by Staff-holons.

The PROSA architecture is used for instance to identify agents in the manufacturing control system developed in the ESPRIT LTR project MASCADA [Peeters et al., 1999]. The entities comprising the lowest level of the architecture in the example application of this thesis (Chapter 4) are also specified according to PROSA.

In domains other than manufacturing control, a functional identification of agents in an application is an alternative to entity-oriented partitioning. The RETSINA agent architecture [Sycara et al., 1996], for instance, comprises three abstract types of agents: interface agents, task agents, and information agents. In contrast with the entity-based agent types in PROSA, these agent types represent functions to be performed by the system.

An example for the function-oriented partition of an application on the basis of the RETSINA architecture is the portfolio management system WARREN [Sycara et al., 1995]. Task agents in WARREN are the “Fundamental Analysis Agent”, the “Technical Analysis Agent”, or the “Breaking News Agent”. Each agent represents a particular function of the system. New functions are added or removed by changing the agent structure.

Applying the function-based partitioning to manufacturing control applications, functions like mid-term planning, short-term scheduling, transportation, processing, or quality checking may be identified with agents. Such an approach tends to result in hierarchical systems that are similar to the traditional ones (Section 2.2.1) except that there may be more autonomy in the modules.

#### *Interaction Structure*

The identification of the agents in a distributed manufacturing control architecture influences the resulting structure of the interactions. Following [Parunak et al., 1997], there exist four different ways of structuring agent interaction in manufacturing architectures.

The simplest form of interaction configuration is given in broadcast communication. Examples for broadcast communication in manufacturing control systems are force fields as applied in [Vaario and Ueda, 1998], or the undirected communications in LMS [Fordyce and Sullivan, 1994] or A-Teams [Lee et al., 1996] [Akkiraju et al., 1998]. The MASCADA application architecture [Peeters et al., 1999] combines broadcast structures (pheromone fields) with direct interactions along the process flow.

Interactions structured along the information and material flow are predominant where the agents map to entities in the manufacturing system. The AARIA system [Parunak et al., 1997] and the WEST system [Bussmann and Schild, 2000] are examples for process flow

communication paths in manufacturing control systems, while ISCM [Fox et al., 1993], for instance, follows the same approach in a business process model.

Process flow structured interactions often employ the contract net protocol [Shaw and Whinston, 1988]. It specifies a sequence of auction announcement, bidding, bid evaluation, and winner announcement. In control systems that are partitioned into manufacturing entities, either the job or the resource may announce (host) an auction. There are resource driven auctions where a resource starts the protocol when it is idle and available (e.g., [Ferguson et al., 1988] [Shaw and Whinston, 1988] [Ramaswamy, 1995] [Dewan and Joshi, 1998]). The announcement is answered by interested jobs, which submit bids for time slots on the resource. Other systems follow a job-driven approach (e.g., [Malone et al., 1983] [Upton, 1992] [Brueckner, 1997] [Bussmann and Schild, 2000]), and there are mixed approaches where jobs and resources host auctions depending on the current load of the system.

Manufacturing control systems that follow a hierarchical approach in the identification of their components (e.g., YAMS [Parunak, 1987], BOSS [Hynynen, 1989] display interaction structures that are hierarchical too. In such a system constraints are communicated top down from higher-level agents to lower-level ones. At the same time status information is sent up from lower levels in the hierarchy.

Mediator-based approaches promote a hybrid of hierarchical and process flow configuration (e.g., [Burke and Prosser, 1994] [Butler and Ohtsubo, 1992] [Interrante and Goldsmith, 1998] [Shen and Norrie, 1998]). These systems identify their primary set of agents according to the entities in the domain. Then, the agents are split into groups and a "mediator" agent joins each group. It is the task of the mediator to oversee its group of agents and to resolve conflicts arising in the interactions of the agents in the group. These mediators often employ centralized conflict resolution mechanisms.

The direct interactions specified in the manufacturing control system presented in this thesis (Chapter 4) are predominantly structured along the process flow, while the hierarchical information exchange is based on stigmergy and may be identified as broadcast communication.

***Interaction Contents***

Agent interactions in manufacturing control systems are also differentiated according to the contents of the messages. In general, there are symbolic and numerical contents.

Symbolic messages have their roots in classical artificial intelligence research. They often convey complex status and constraint information, which in turn requires considerable intelligence on the side of the receiver to parse and process such a message sufficiently. As a consequence, the more complex the contents of the messages, the more complex the agents tend to be (e.g., the agents in FAKOS [Baumgärtel, 1999]. But there are also exceptions to such a tendency. The WEST system [Bussmann and Schild, 2000], for instance, requires only limited reasoning capabilities in the individual agents.

Numerical messages, on the other hand, do not require complex agent reasoning except for very specific tasks. Natural or economic models often inspire agent systems interacting on the basis of numerical messages. In numerical message exchange balanced and unbalanced exchanges are considered.

Balanced numerical interactions are exemplified in market systems. These systems conserve a currency in its internal trades. The "money" an agent earns is the same amount

another agent has to spend. Examples of market-based systems in manufacturing control are found in [Baker, 1996] [Lin and Solberg, 1992] [Upton et al., 1991] [Walsh et al., 1998], [Odell and Greenstein, 1999] or [Parunak et al., 1997].

In unbalanced numerical interactions, the messages are independent of the agents. As a consequence more than one agent may read the communicated information since the message itself is not erased when it is received. Force fields [Vaario and Ueda, 1998], posted numerical priorities on a blackboard [Sikora and Shaw, 1997], or pheromone fields [Parunak and Brueckner, 2000] are examples for such interaction contents.

The manufacturing control system presented in this thesis combines unbalanced numerical interactions (via pheromone fields) and simple symbolic interactions along the process flow.

### 2.2.4 Summary

Bussmann and McFarlane present the following vision of holonic manufacturing control:

*"In the beginning, a holonic manufacturing system only consists of a set of unorganized resource holons which form a manufacturing holon. Upon arrival of an order, however, the manufacturing holon creates an order holon which starts to negotiate with resource holons on the provision of certain manufacturing operations. During the negotiation process, the order holon demands specific properties of the operation, such as high quality or high throughput, while the resource holons try to maximize their utilization. At the end of the negotiation, the resource holons move to form the agreed manufacturing line and the order holon initiates the creation of work piece holons.*

*The work piece holons enter the manufacturing holarchy (e.g., from the stock) and immediately bargain for resources in order to get processed. Each work piece holon does so individually and focuses on the next operation(s). Once these operations have been performed at a resource, the work piece re-initiates the bargaining with the remaining (next) operations. The overall organization of the resource holarchy - initially or subsequently negotiated between order and resource holons - assures that the work piece load is efficiently distributed over the available resources in order to achieve the global goals of this holarchy.*

*In case of a disturbance, the affected resource holon removes itself from the resource holarchy and goes to a repair booth. The remaining resource holons re-organize themselves in order to account for the capacity loss. From the point of view of the work piece holons, the processing continues as usually, only with less resource holons with which it may bargain. After repair, the resource holon tries to join the resource holarchy again.*

*At the end of the order processing, the order holon is removed and the resource holarchy dissolves into the resource holons which then try to participate in new order holarchies."*

—[Bussmann and McFarlane, 1999]—

Following the holonic vision, this thesis demonstrates the design and evaluation of a complex stigmergetic synthetic ecosystem in the example of a manufacturing control system. The spontaneous self-organization of agents in the processing of an order to the

system is coordinated in sign-based stigmergy where the signs are pheromones managed by an extension to the software run-time environment of the agents – the pheromone infrastructure.

The presented manufacturing control system goes beyond the holonic vision. The emerging patterns in the material flow are observed by a higher-level advisory system, which considers the manufacturing process under global performance goals. On the basis of the observation the self-organized processes on the entity-level are influenced. The presented application architecture explicitly incorporates emerging system features.

# Chapter 3 - Adopting Nature's Principles

*Chapter 3 first states a number of guidelines for the design of synthetic ecosystems. On the basis of these guidelines and the observation of coordination mechanisms in insect colonies, an extension to traditional agent runtime environments, the pheromone infrastructure, is introduced. In the remainder of the chapter the pheromone infrastructure is formally modeled and an agent-oriented implementation is specified.*

## 3.1 Synthetic Ecosystems in Manufacturing Control

The synthetic ecosystems (SE) approach proposes design principles applying to the design of agents and of the architecture of the agent system as a whole, and it suggests principles for the interaction design. The distinction is not a "clear" one, but for each principle the primary focus may be identified.

In the following each principle is stated, motivated, and explained in its effects. Where applicable, consequences for the design of agent-based manufacturing control systems are discussed. Then, returning to the emergent food foraging behavior of an ant colony, the application of the principles in the "design" of a natural agent system is shown. Finally, Section 3.1.5 describes a generic software infrastructure for artificial agents that supports the application of the design principles in the development of self-organizing multi-agent coordination.

The design principles have several sources. Some principles are already suggested in [Kelly, 1994] or [Parunak, 1997]. Others occur in a wide variety of complex systems that are examined in [Flake, 1999], [Kauffman, 1993], or [Bonabeau et al., 1999]. Finally, there are design principles that are added to the list simply because they represent sensible approaches to the design of complex software systems for real-world applications.

### 3.1.1 Single-Agent Principles

**Things, not Functions (SE-Principle 1)**.—*Identify agents on the basis of existing physical entities, rather than based on functions performed*.

The agent system is assumed to directly control parts of the physical world. If the agents are identified according to distinct physical entities, the responsibilities remain well defined and knowledge and skills are encapsulated. Furthermore, when agents are mapped to real-world entities, the agent interactions may be adopted from their real-world counterparts. The physical world provides the relevant trigger-events, advancing agent interactions. A similar approach is promoted in object-oriented design.

Modeling agents on the basis of physical objects is intuitive. Changes in the composition of the controlled system are easily handled by changing the composition of the agent system. But automatic re-configuration mechanisms are still required.

In a manufacturing control environment, physical entities that could be mapped to agents are, for instance, workpieces, machines, tools, workforce, or transport elements. What granularity is finally chosen for the agent model primarily depends on the applied

control mechanisms.

Functions are only mapped to agents wherever legacy systems or watchdogs are part of the resulting control system. Legacy systems in manufacturing are often encountered in MRP(-II) or quality management systems.

**Small Agents (SE-Principle 2)**.—*Keep agents of a synthetic ecosystem "small". The effort in communication and computation to execute an agent in a runtime environment must be small compared to the execution of the multi-agent application. Furthermore, there must be only a small impact from the activities of an agent on the state of the whole system.*

Modeling a system on the basis of small agents has several advantages. Small agents are easy to implement, to debug, to understand, and to change. A software system made up of many small agents of several types is easier to implement and stabilize than a system made up of just some complex agents. With clear-cut responsibilities of an agent requiring the repeated application of only some agent skills, the challenge of engineering the agent itself is small.

A system of many small agents may easily express a wide range of very complex behavioral patterns. Programming all these patterns into one central entity requires a larger effort in implementation, debugging, and maintenance. Thus, the state-space covered by a system with an equal effort in implementation, stabilization, and maintenance is much larger for systems made up of small agents.

In general, if an agent is small in its impact on the whole system then its failure should also have a small impact only. But, while a single agent may fail without major consequences for the system, an error in the software implementation of an agent type still is critical because there are potentially many agents carrying the error.

Keeping agents small is also achieved through a restricted life cycle for each agent. Thus, aggregation of minor faults into a major one in the state of an agent over time may be avoided. The systematic death of agents is also a way to provide forgetting in the system (SE-Principle 16).

Modeling agent systems for manufacturing control based on small agents obviously influences the granularity of the model. For instance, instead of modeling whole manufacturing cells as one agent, a cell may be controlled by an ecology of agents, one for each part of the cell. The level of abstraction gives another important influence on the granularity, in which the control system operates. The focus may be on separate workpieces or on aggregated material flows, for instance.

**Diversity, Heterogeneity (SE-Principle 3)**.—*Have many agents of different types (diversity) and introduce slight variations within the agents of one type (heterogeneity) either sequentially (generations) or in parallel (families).*

Diversity in multi-agent systems supports the notion of small agents. With each agent being small, there have to be different agent types with different responsibilities, skills, and knowledge. Together, agents of different types perform complex tasks in a coordinated manner.

Heterogeneity, on the other hand, enables the system to try out different approaches in solving the same problem. Heterogeneous agent types are more adaptive than homogeneous ones. Providing that effective use of heterogeneity is made, the resulting system behavior may maintain behavioral patterns to fulfill its tasks in a changing

environment.

Heterogeneity always poses a certain risk to the system. When trying out another approach to a problem, it is not guaranteed that the solution found is an optimal one. Not even feasibility is guaranteed in all cases. A sub-optimal resource usage results. Hence, the system has to trade off exploitation of known solutions against exploration for new solutions. The level of risk taken is another factor differentiating the agents.

**Redundancy (SE-Principle 4)**.—*Introduce redundant elements in the synthetic ecosystem either by tackling the same problem in parallel by multiple agents or by enabling the agents to take up tasks, other agents have not been able to finish for some reason.*

Having redundant elements in the system is advantageous for the stability of the system. If an agent fails, then the system does not have to break down as a whole. Instead other solutions are instantly available or other agents pick up the tasks of the lost agent automatically.

Redundancy also requires additional resources. Either, there are more agents operating than minimally required. Or, the effort in the design of the agents is increased. Again the need for backup in case of local breakdowns has to be balanced against the costs for the additional resources.

### 3.1.2 System-Architecture Principles

**Decentralization (SE-Principle 5)**.—*Decentralize the synthetic ecosystem consequently and design its structure inherently dynamic. The agents have to explore the system by themselves, encountering other agents as source of information or services. Centralized services provided by the environment or by specialized agents are not permitted.*

Decentralization is a major principle in the agent-oriented approach in general. But many applications still retain central elements in their structure. The need for a global White Pages (WP) service is often an indicator for structural centralization. There is someone who needs to know all agents.

Consequent decentralization prevents the occurrence of bottlenecks in computation or communication when the system is scaled up. The multi-agent system remains truly open since each agent has to operate as if it just had joined in. Teams are dynamically formed to fulfill arising tasks, and when a task is fulfilled the team falls apart.

To enforce decentralization, a global WP service should be avoided. Agents should learn about potential partners for interaction from other agents only. References to agents are kept only as long as they are actually used in an interaction. Keeping stale references is a risk to the stability of the system, since agents may join in or leave the system at any time.

Even strictly local WP service is permissable along the scaled dimension only. In manufacturing control applications such a dimension may be the physical space. All agents that are located in a spatial structure may be listed in local White Pages. A local WP service comprises references to agents that are all located in the same local area.

When the system is scaled up, the space covered by the system is extended. With only local WP services available, the number of agents in existing local WP is not increased.

Rather, new WP clusters for the new space are created.

**Modularity (SE-Principle 6)**.—*Grow a synthetic ecosystem by adding parts, with each part encapsulating some functionality provided by an ecology of agents. The system is designed, set up, tested, and started in a given sequence of added parts, starting with a minimal configuration.*

Ideally the parts of the system are strictly layered. Each layer extends the functionality of the layer beneath it. Every step in the design adds another layer to the system from bottom up.

There is a number of parts forming the core of the system – the minimal configuration. The core has to guarantee the basic operation. All hard constraints set to the operation of the system have to be fulfilled by the core already.

Additional parts of the SE may be added or removed dynamically, minding the interdependencies among the parts. In a layered structure, for instance, higher layers generally require the presence of some or all layers beneath for correct operation. Each additional part refines and optimizes the operation of the whole system. While the hard constraints have to be fulfilled in any valid configuration, specific additional parts fulfill optional requirements or goals.

In manufacturing control applications the core parts have to realize the manufacturing of products. They guarantee that every workpiece is processed correctly. No workpieces are ever taken to machines, to which they must not go. Only the required processing steps are performed and the processing sequence, which is constrained by the process graph of a product, is minded.

Parts extending the core behavior could realize fairness in processing or avoidance of deadlocks in transport. The performance of the manufacturing process is improved according to the production goals or additional services are provided to the operators of the system.

**Parallelism (SE-Principle 7)**.—*Solve multiple problems in parallel as they arise in different groups or teams of agents that are created dynamically. Enable the agents to participate in several teams concurrently, internally intertwining their interactions within the different teams.*

The agents are very specialized in their skills. There is almost never an agent able to perform a task alone. In a consequently decentralized and asynchronous system, tasks arise in parallel. Hence, there must be more than one team operating at the same time. Agents with specific skills or knowledge may join multiple teams. These agents have to be able to manage multiple interactions in parallel.

Parallelism occurs in manufacturing control systems. Different workpieces have to be processed at different processing resources at the same time. Agents controlling these resources solve these tasks in parallel. Depending on the design of the agent system, there could be agents providing process information about the manufacturing of a product. These information providers then possibly join multiple teams in parallel.

**Bottom Up Control (SE-Principle 8)**.—*Control the physical environment from bottom up through the interactions and decisions of a multitude of agents. There must be no single entity, to which control can be pinpointed. With many (and small) agents wielding control of the environment, emergent features should dominate the design of the system.*

Control emerging from bottom up is another consequence resulting from strict decentralization. In a synthetic ecosystem there are some agents that are most closely linked to the physical environment. These agents have the primary control of the physical entities. Since the agents are identified according to things, not functions, there is no centralized control of the environment. Instead, the global control emerges from the local activities of the agents.

Other agents, conceptionally farther away from the physical world, influence the primary controllers. The emerging control pattern changes. With each component added to the system, the complexity of the resulting behavior controlling the physical world grows.

A single agent does not have much impact on the operation of the system. The design of the behavior of the agents of a specific type defines the global behavior. As a consequence of the multiplicity of the agents and their repeated actions and interactions small changes in the specification of an agent type might result in the emergence of completely different system-level behavior.

### 3.1.3 Interaction Principles

**Locality (SE-Principle 9)**.—*Restrict the sensor input and the direct effect of actions of an agent to its local environment. No agent must have access to all information currently available. No agent may affect major segments of the controlled physical world or interact directly with agents outside its local environment.*

Along with the principle of keeping agents small, remaining local is the most important principle in the design of a SE. It is introduced on the basis of results gained in research of complex dynamical systems. Examining random Boolean networks [Kauffman, 1993] for instance, it was concluded that the connectedness of the local entities strongly influences the emerging global behavior. If too many agents influence other agents, then ordered system-level behavior is hard to achieve. On the other hand, the available system behavior is strongly reduced if there are too less influences.

To reduce the connected-ness of the agents in the SE, each agent only perceives and influences a small subset of the overall system. If there is a spatial structure imposed on the agent system, the access of an agent is reduced to its local neighborhood. The restriction goes hand-in-hand with the restriction of White Pages services to only local ones. The availability of only local WP services effectively disables any system-wide communication in dynamically structured agent systems.

In manufacturing control applications, locality enforces local team formation linked to the arising tasks. Transport is affected even more by the design principle. If the locality restriction is observed, then the routing of workpieces cannot be handled by a global transport system. Rather, subsequent local routing steps on the basis of locally available information only should be taken.

Additionally, process information for the manufacturing of products should not be globally available. It must be localized specific to the respective section of the manufacturing system.

**Indirect Communication (SE-Principle 10)**.—*Wherever feasible, decouple agents in their communications, transferring information through a shared environment instead of directly passing messages.*

Message-based exchange of data among agents is one way to provide information required in the decision processes. Another one is to transfer data indirectly through the environment of the agents.

Direct agent-to-agent communication requires the partners in the exchange to synchronize. One partner requests information and the other partner now is forced to handle the request, regardless the current priorities of the agent. The requesting agent is forced to wait until the provider finally has come up with the data. If the agents operate on different time scales, then the whole exchange is extremely difficult to handle, especially when the provider of information is the slower one.

Another disadvantage of direct agent-to-agent communication is the effort required to stabilize such interactions in the face of possible agent-breakdowns. Complex transaction mechanisms have been developed to make sure the data is really transferred, and backtracking mechanisms allow resetting an interaction in case one of the partners has broken down.

Finally, aggregation of data from many sources for the use of multiple agents (many-to-many communication) takes more effort using direct interactions. Either, there would have to be a specific service provider aggregating the data, and each recipient would have to request the result of the aggregation from the provider. Or, the recipients do the aggregation on their own.

Indirect communication requires an environment jointly accessible by all potential senders and receivers. It must be possible to store and retrieve data in such an environment. The provider of information stores the data it has generated in the environment. The data is picked up by anyone requiring it. Stigmergy (Section 2.1.6) is based on indirect communication.

Indirect transfer of data requires no explicit synchronization. The agent that needs information only has to wait if there is no data generated yet. Otherwise it just reads it from the environment. The information provider generates the data when it fits into its own priorities.

Since there is no direct coupling of agents in the data transfer, the agent activities are more stable. Neither the information provider, nor the consumer has to care if the other breaks down. They do not have to perceive each other at all. But nevertheless, the internal agent operation has to prepare for missing information.

Most aggregation processes still require an agent to execute them. The agent may pick up stored data from the respective providers and after processing it stores the result. Each agent that requires the aggregated data just takes it. But, depending on the chosen active environment, some processes may even be executed automatically.

In manufacturing planning and control there are many processes executed at different time scales, with data being exchanged among these processes. For example, there are the different planning processes ranging from long-term to short-term planning. Each planning process takes the results of the previous one and refines it. Continuous planning and replanning mechanisms should be based on indirect interactions to avoid the coupling of processes on different time scales.

**Recursion, Self-Similarity (SE-Principle 11)**.—*Repeat structurally similar interaction mechanisms on different levels of the agent system (recursion). Wherever appropriate, apply a successful approach in solving a problem to similar problems at all scales (self-similarity).*

Recursion and self-similarity are applied all over nature. Many large systems owe their complexity the recursive repetition of simple mechanisms or structures. Self-similar structures are found in the growth of all kinds of plants, in cells, organisms, or ecosystems.

Structural recursion provides simple building blocks that fit together to form larger blocks. It was already suggested that a SE might be modularized. When applying recursion to the structure and the mechanisms of the parts, modularity becomes simpler to realize and the resulting system is much easier to understand.

A very obvious example for recursion in structure and mechanisms in manufacturing execution is found in the AMIS system [Odell and Greenstein, 1999]. The agent responsible for the fulfillment of the task breaks down an incoming task into sub-tasks. The agent either performs the sub-tasks, or they are sold out to other agents. These sub-contractors repeat the process recursively. The general principle is very simple, but the emerging global behavior is sufficiently complex and dynamic to fulfill very complex orders.

**Feedback, Reinforcement (SE-Principle 12)**.—*When coordinating large crowds of small agents turn to feedback and reinforcement effects. Always take into account the feedback of effects of actions in the environment back into the agents' operation.*

An important principle in the design of emergent coordination mechanisms in a SE is the focus on increasing returns. Knowledge, skills, or behavioral patterns that produce positive results in agent activities are reinforced in their importance with every usage. The consequence of the reinforcement is a higher impact on future agent behavior. Thus, cause and effect in agent activities are linked in a positive feedback loop. The design principle holds for the single agent as well as for groups of agents.

Another form of feedback inevitably occurring is caused by the activities of agents in their environment. Agents change their surroundings by their actions. Hence, future actions must take these changes into account. Complex cause-and-effect chains couple the agents. Most of these chains feed back into the originator of the effects. Such feedback through the environment must be taken into account when designing the agent behavior. When effectively used, feedback-aware behavior may drastically improve the performance of an agent.

Manufacturing control applications may make use of increasing returns effects. For example, the selection of a processing resource from a set of available ones should increase the probability for the following agents to chose the same resource as well, if the processing was successful. Such a mechanism may be used throughout the production system and similarly for the selection of transport routes.

**Randomization (SE-Principle 13)**.—*Introduce a random factor into the moment and the outcome of agent decisions in massively parallel interactions of many agents to prevent the emergence of negative synchronism and to explore alternative solutions.*

There are devastating effects in multi-agent coordination known as thrashing, stemming from synchronism in parallel interactions. They occur when many agents take a deterministic decision regarding the same choice at the same moment based on the same information. The classic example for thrashing is the synchronized selection of the shortest queue, when many agents rush from one queue to another because they all select the same queue, making it a much longer one. Adding randomization to the decision moment and to the outcome of the selection of the agents effectively prevents thrashing in the example.

Additionally, randomization may dampen avalanching effects caused by feedback. The

number of agents joining an avalanche is reduced, if their decision is not deterministic. Randomization provides alternatives that may prove advantageous to the system as a whole.

Inherent randomization of decision processes prevents the system from settling in at a local optimum. It is kept alive and adaptive, always trying out slight variations of the main theme. Simulated annealing technique is a successful application of randomization in search mechanisms.

Randomization breaks symmetries and thus supports self-organization in complex dynamical systems. With non-linear processes in such a system, very small random fluctuations may be amplified and one of the many stable states of the system is selected.

In manufacturing control applications following the SE-approach, many decisions are taken in parallel and independently by many agents. Without randomization, such massive parallelism is hard to handle and stabilize. The system is prone to catastrophic thrashing and avalanching. Feedback may not be used in coordination; instead it would have to be prevented by all means.

**Evolutionary Change (SE-Principle 14)**.—*Always prefer gradual or evolutionary change in the state of the system to abrupt or revolutionary change. Global evolutionary change may be achieved if each agent changes gradually only.*

A situated agent system acts in an external environment, which has its own dynamics. All actions in the environment take a while to become perceptible in their consequences to the agent system. Repeated abrupt changes in the state and in the behavior of the system may superimpose potentially contradicting actions in its environment.

Abrupt major changes in the state of a system are in their consequence comparable to longer jumps in the search within weakly auto-correlated fitness landscapes. There is no prediction of what the resulting fitness – the resulting performance of the system – will be after the change [Kauffman, 1993]. How timid the system should be in its change is dictated by the dynamics of the surrounding environment, as the optimal length of jumps in dynamic fitness landscapes depends on the ruggedness and the strength of the auto-correlation of the landscape and on the dynamics of its change.

Evolutionary change is further supported by the demand for locality. Each agent perceives and acts in its local environment only. To initiate and control major changes takes a huge effort. It is much easier to propagate changes in a distributed manner. Depending on the granularity of the system, propagation of change takes a while to affect the whole system. Hence gradual change is achieved.

**Information Sharing (SE-Principle 15)**.—*Include information sharing mechanisms into the standard operation of the agent system. Share information between generations of agents; learn as individuals, or as a society.*

Whenever a new approach to solve a problem is tried out, the agents put resources at risk. If the experiment fails, the invested resources are lost. A successful trial yields a new solution. To make maximal use of the invested resources, the new information should be shared among the agents.

Information sharing may be incorporated into the activities of the agents wherever appropriate. An effective way to make sure information is shared is to underpin the information storage infrastructure with automatic information sharing mechanisms. Information stored by an agent is potentially made available to others, always preserving

locality.

As already mentioned in the discussion of randomization (SE-Principle 13), there are many opportunities for a manufacturing control system to learn and to adapt. Automated learning and adaptation processes may help to simplify system design.

**Forgetting (SE-Principle 16)**.—*Include forgetting if an agent or the agent system is enabled to learn. The multi-agent system should include mechanisms for automatic dissipation of acquired information over time.*

Without forgetting, information remains within the system even when it is outdated. The agents would have to make sure that data that is no longer valid is removed. The removal becomes a major effort if information is stored in the environment for indirect communication.

When reinforcement mechanisms are applied, automated forgetting may prevent overload. Without a process that levels out the bias values for a probabilistic selection from a set of options, repeated reinforcement could take these values beyond any sensible limit.

The loss of information helps the system to adapt in a similar way randomization does. Reinforcement processes tend to bias the agents, and hence the whole system, towards specific solutions or behavioral patterns. The bias is intentional since these solutions have proven themselves in the past. But, in a changing environment, good solutions are never sure to remain good for all time. To prevent the system from being trapped in one solution and to remain adaptive, acquired information must be given up.

**Multiple Goals (SE-Principle 17)**.—*Before designing the agent system, identify the different goals and their respective type. Design the system to be able to handle multiple goals.*

The operation of synthetic ecosystems in real-world applications almost always requires the fulfillment of multiple goals. Furthermore, these goals are mostly maintenance-goals without a final goal of the operation of a system. Instead, the agent activities maintain an ongoing operation and adjust to cope with disruptions. Often, a number of given requirements are sufficed.

The continued sufficing of maintenance-goals instead of the optimized fulfillment of achievement-goals is one of the major differences between the operation of a SE in (manufacturing) control applications and systems in problem solving domains like process planning or failure diagnosis. Before the design process is started it has to be clarified what problem class the system will have to cope with. In the case of achievement-goals, a predominantly SE-oriented approach may even be inappropriate.

If the application is dominated by maintenance-goals, then traditional problem-solving techniques are still applicable in the performance of specific sub-tasks. The fulfillment of all requirements over the full run of the system is verified in the evaluation of the global performance. To compare the performance of systems, parameters like resource usage, efficiency, or the fulfillment of other soft requirements are evaluated.

In many mass-production systems in industry the predominant maintenance-goal is high global throughput, followed by high resource usage, low work-in-progress, or low production costs. These goals are contradictory and quantitative; hence they can only be sufficed.

### 3.1.4 Go to the Ant

The design principles are identified and illustrated in the context of a single task of a natural agent system. Returning to the ant colony visited already in Section 2.1.6, the joint food foraging is considered. This coordination mechanism is probably the most cited example of naturally occurring self-organization in literature related to swarm intelligence. A "(#)" references the number of a design principle when identified in the description.

Social ants construct networks of paths that connect their nests with available food sources. Mathematically, these networks form minimum spanning trees. Thus, they minimize the energy they spend to bring food to the nest. The colony accomplishes this feat without any centralized control. The paths emerge from the activities of the individual ants (8).

A single ant is very small compared to the whole colony in terms of resource usage and impact (2). As a physical entity in the environment (1) it has a spatial location. Its sensing and acting is restricted to a very small environment (9, 5). In a colony there are different types of ants fulfilling specialized tasks (3). Following natural evolution, recombination and mutation effects ensure heterogeneity in the types.

Many ants attempt to solve the food-foraging problem in parallel (4). Depending on the current situation in the environment of the colony, some ants may succeed while others fail. If the rate of failures is kept below a certain threshold, the system remains stable.

Ants communicate indirectly through their environment (10). Direct ant-to-ant communication occurs only occasionally. Highly volatile chemical substances, called pheromones, provide the means for indirect communication. Each ant is able to produce pheromones. It drops them to the ground or it sprays the pheromones into the air. At their current location, ants perceive the different pheromones separately. They discern the local strength and the local gradient of the strength of pheromones.

Pheromones are carriers of data, since each existing pheromone type is deposited in a specific context. Once dropped to the ground, pheromones of the same type aggregate in strength. The aggregated strength is available to all ants present. The perceived strength and gradient is incorporated into the probabilistic decision processes of the single ant.

Pheromones evaporate over time. The continuous decreasing of the local pheromone strength is governed by a dispersion function of the form $s(t+1)=s(t)*E$, where $E$ is a constant evaporation parameter between one and zero specific for each pheromone type. Each ant foraging for food expresses a behavioral pattern, which is emulated with sufficient correlation by the following set of simple rules that incorporate one specific pheromone type.

**Rule (1)**.—Avoid obstacles.

**Rule (2)**.—If you find yourself at food and you are not holding any, pick the food up.

**Rule (3)**.—If you find yourself at the nest and you are carrying food, drop the food.

**Rule (4)**.—Select every step from the current distribution over possible directions. If no pheromones are sensed, use a uniform distribution, executing Brownian motion. If pheromones are sensed, bias the probabilistic selection in favor of the scent's direction.

**Rule (5)**.—If you are holding food, drop fixed amounts of pheromone at a constant rate as you walk.

The presented behavior of the food-foraging ants falls into different parts (6). At the core there is obstacle avoidance (Rule 1), making sure an ant can navigate even in complex landscapes.

The second building block of the food-foraging behavior makes sure the food is picked up and delivered (Rules 2, 3). Actually, the food-dropping rule (Rule 3) is more complex in reality, where food is stored at specific places of the nest.

To ensure the whole surroundings of the nest are covered by the search of the colony for food, random search is executed. Rule four provides that even in the absence of any additional knowledge the ants eventually find a food source, if there is any. Random search is the ultimate "weak method" in navigating through a search space. By itself, it breaks down due to the combinatorial explosion. But combined with the information sharing realized by the pheromones, it keeps the system robust, adaptive, and alive.

In the presence of pheromones, the walk of an ant is still randomized. But the selection of the direction is weighted in favor of the direction of the scent. The principles stated in the SE approach require randomization (13) of the moment and of the outcome of the internal decisions of the agents wherever possible.

The probabilistic component in the movements of an ant realizes an ongoing search for improved solutions to the food-foraging problem. As a consequence of rule five, new paths found in the search are shared with other ants (15). The evaporation of pheromones realizes forgetting (16), which is required to "forget paths" to depleted food sources. Furthermore, the gradual change of the information incorporated into the decision processes depend on may lead to an evolutionary change of the state of the system (14).

An ant regularly drops pheromones while it carries food. Thus, trails are only created by successful ants, which have found food. Additionally, each successful ant that finds a pheromone trail in its random walk follows the trail to the nest. Thus, positive feedback occurs (12). Negative feedback comes into play with the restricted number of ants and the finite amount of food.

Finally, the system as a whole – the ant colony – suffices a number of maintenance-goals (17). Besides foraging for food, it has to sort the brood, and build and maintain the nest. The colony has to protect itself and eventually it spawns new colonies. Each task is fulfilled as good as possible without achieving a final solution. The colony maintains and adapts its global behavior to constantly suffice the set goals in a changing environment.

### 3.1.5 Return from the Ant

Consider a synthetic ecosystem where software agents act in a physical environment. On the one hand, the agents control physical entities in the real world. But on the other hand, there are interactions among the agents in a software environment.

To enable indirect coordination among software agents in the same way social ants coordinate, the software environment should emulate the "services" provided by the real world to the ants. The part of the software environment realizing the services is called the pheromone infrastructure (*PI*). In the following, major features of the *PI* are specified, using terminology taken from descriptions of insect colonies.

The *PI* models a discrete spatial dimension. It comprises a finite set of places and a topological structure linking the places. A (topological) link connecting two places has a

downstream and an upstream direction. Thus, for each place there is a set of downstream and a set of upstream neighbors – places that are directly linked to it.

Each agent in a SE is mapped to a place. The mapping represents the current location of the agent, which may change over time. A change in the location of an agent must always follow a link in a downstream or upstream movement.

The *PI* models a finite set of pheromone types. A pheromone type is a specification of a software object, comprising a "strength" slot and other data-slots. The domain of the "strength" slot is the set of real numbers. The domain of every other data-slot must be finite. For each pheromone type, a propagation direction (downstream or upstream) is specified.

The *PI* handles a finite set of (software) pheromones for each pheromone type. A pheromone is an incomplete specialization of its respective pheromone type. Every data-slot, except "strength", is assigned a value from its respective domain to form one pheromone. A pheromone provided with a "strength" value is interpreted as a specific amount of the pheromone. All possible pheromones of a SE may be stored together with their respective local amount in the pheromone management of each place. In the following, the term "strength of a pheromone" is used as shorthand for the value in the "strength" slot of a pheromone.

An agent may perform the following activities at its current place in the *PI*:

- Access the references to all agents located at a place.
- Perceive the downstream or upstream neighbors of a place.
- Sample the local strength values of a specified set of pheromones.
- Initiate a change in the local strength of a specified pheromone by a specified value.

The *PI* manipulates the values in the "strength" slots of the pheromones at each place in the following way:

**External Input**.—Based on a request by an agent, the strength of the specified pheromone is changed by the specified value.

**Internal Propagation**.—Assuming an external input of strength *s* into a pheromone *g* at a place *p*. The input event is immediately propagated to the neighbors of *p* in the propagation direction of *g*. There, the local strength of *g* is changed by an input weaker than *s*. An even weaker input propagates to the following neighbors. The stepwise weakening of the input is influenced by *g*'s propagation parameter.

**Evaporation**.—Without taking changes caused by external input or propagation into account, the strength of each pheromone is constantly reduced in its absolute value. The reduction is influenced by the evaporation parameter of the pheromone.

There is a major difference between the algorithms realized in the *PI* and those observed in nature. After an ant places pheromones on the ground, evaporation disperses it. Particle by particle the pheromone moves through continuous space driven by Brownian motion. At the initial location the amount of pheromones is reduced while it builds up somewhere else or vanishes completely.

In the discrete space of a *PI*, propagated pheromones have only specific locations on which to "settle down". Furthermore, the structure of the space is not homogeneous – at some places pheromones may be propagated to many neighbors, while at other places no further propagation is possible. Finally, in an implementation of the *PI*, continuous erosion

of the pheromone levels would require interaction among all places at all time.

As a consequence, the mechanisms of evaporation and propagation of pheromones are modeled separately in the software environment of the agents. Instead of continuously exchanging particles among the places, there is one "wave" of input events running along the links, which is triggered by the original input of an agent.

A pheromone infrastructure realizes an application-independent support for a SE designed according to the proposed principles. Specifically, a *PI* supports:

Decentralization (SE-Principle 5).—The mapping of each agent to a place in the *PI* introduces a spatial structuring of the agent system. With the ability of each agent to request the references to other agents located at its place, a local White Pages (WP) service is realized. The topology data permits the agents to explore the spatial structure of the system.

Locality (SE-Principle 9).—Each agent is only permitted to request information local to its current place. It can perceive the agents located at its place, the neighboring places, and the values of the "strength" slot of the pheromones present at its place. It can act only on local pheromones and it can interact directly only with local agents whose references it receives through the local WP service.

Indirect Communication (SE-Principle 10).—A pheromone represents specific data in the same way the pheromones of the ants do. If one agent changes the value in the "strength" slot of a local pheromone, other agents can perceive the change. Thus, information is transmitted through the environment. The environment aggregates data automatically when multiple inputs to the same pheromone occur. The pheromone infrastructure directly supports sign-based stigmergy.

Feedback and Reinforcement (SE-Principle 12).—The pheromones provide the means to externally store and reinforce bias values for the decision processes of the agents. The automatic evaporation of the pheromones requires a continuous reinforcement of stored information.

Randomization (SE-Principle 13) and Evolutionary Change (SE-Principle 14).—Wherever the value in the "strength" slot of a pheromone at a place is incorporated into the decision processes of the agents, randomization occurs because the agents operate asynchronously and the decision parameter changes over time independent of the activities of the agents. An even better way to realize randomization is to have the strength directly determine a probabilistic weighting of decisions.

Information Sharing (SE-Principle 15).—Putting data into a *PI* provides the ideal means for information sharing among the agents. As the pheromone trails in the food-foraging example represent the acquired knowledge of the colony of paths to food sources, the data in a *PI* represents the accumulated knowledge of a synthetic ecosystem. It is stored for the agents to retrieve locally, and it is even propagated to other places to make information available in a wider area.

Forgetting (SE-Principle 16).—A *PI* provides an automated way of forgetting information stored there. Data put into a *PI* evaporates over time. If the "strength" slot is taken as an indicator for the relevance of a specific information then the relevance is lost after a while if it is not reinforced. Any bias acquired by the agents has to be repeatedly reinforced to realize a probability distribution other than a uniform one.

If the *PI* is incorporated into the runtime environment and if coordination mechanisms

are implemented that are analogous to the pheromone-based food-foraging behavior of ants, then other principles in the design of synthetic ecosystems are indirectly realized too. Consider parallelism (SE-Principle 7) and the control from bottom up (SE-Principle 8). Following a pheromone-based approach requires the agents to be able to handle parallel interactions, and control decisions emerge influenced by the successive inputs of agents to pheromones. Self-similarity, for instance, is encountered in the regular input to a pheromone as in rule five of the food foraging example. Such a regular input behavior is applicable to a multitude of information transmission scenarios as the following chapters demonstrate.

### 3.1.6 Summary

Section 3.1 proposes a set of design principles for synthetic ecosystems in manufacturing control applications. The principles have been adopted from a general set of SE principles, from research into complex systems, and from previous experience with agent system design. It is shown in the food foraging example that these principles are also found in pheromone-based coordination in nature.

A set of features assigned to a specific part of the software environment of agents is suggested under the assumption that synthetic ecosystems for manufacturing should be designed according to the proposed principles. A "pheromone infrastructure" provides the agents in a synthetic ecosystem with a similar environment as natural agents, like ants, have available to "execute" their stigmergetic coordination mechanisms.

As an application-independent part of the software environment, the *PI* provides all major elements that are required in the design of pheromone-based coordination mechanisms for software agents. But, for a widespread use of such an approach to system design, a deeper understanding of the *PI* is still required.

The principles presented in the synthetic ecosystems approach to manufacturing control and the general use of a pheromone infrastructure in software agent development are revisited in the "Synthesis" section (Section 6.1). There the advantages and disadvantages of the approach are discussed on the basis of the guided manufacturing control system presented in Chapter 4.

## 3.2 Analyzing the Pheromone Infrastructure

In the following, a formal model of the *PI* is specified. The features of the model are analyzed in different scenarios. A real-world example analysis of a pheromone-based control mechanism concludes the section.

### 3.2.1 A Formal Model

The *PI* extends the runtime environment of the agent system of an application. For an analysis of emerging pheromone patterns, agent behavior towards the *PI* has to be formally specified.

The formal model considers one pheromone only. The descriptive model of the *PI* (Section 3.1.5) proposes that different pheromones are managed separately. Input events

for one pheromone do no affect the strength of other pheromones at any place, and also the propagation and evaporation mechanisms do not introduce any inter-dependencies among the different pheromones. Multiple pheromones are considered using multiple models. In difference to the descriptive model, the formal model is based on discrete time.

**Notation 1 (Numbers)**

Throughout the following discussion $\aleph$ denotes the set of natural numbers and $\Re$ stands for the set of real numbers.

The definition of an environment captures the spatial structure of the *PI*. Each pheromone type has a specific propagation direction for the propagation of input events. That means the event of a pheromone deposit at one place is propagated to all neighboring places in the propagation direction of the pheromone type.

The formal model links the places in propagation relations. A propagation relation is analogous to a topological link in the descriptive model that is reduced to the propagation direction of the pheromone. In contrast with the descriptive model, where places are linked bi-directionally and one of the directions is labeled "upstream" and the other one "downstream", the places in an environment are linked uni-directionally. Nevertheless, the term "neighbors of a place" is used to refer to places linked through the propagation relation to a place, even though it is not the traditional bi-directional neighborliness.

**Definition 1 (Environment)**

An *environment* is a tuple $<P,N>$, where

- P is a finite set of places and
- $N \subseteq P \times P$ is an irreflexive propagation relation among places.

The set of neighbors of a place $p \in P$ is defined as $N(p) = \{p \in P : qNp\}$.

The current state of the *PI* model comprises two patterns. For each place the propagation pattern specifies the strength of the propagated input at the place and the pheromone strength pattern specifies the current local pheromone strength at the place.

**Definition 2 (State)**

Let $<P,N>$ be an environment. The *current state of the PI* in $<P,N>$ is given in the two mappings $q$ and $s$, where

- $q : P \to \Re$ is a total mapping that maps a place $p \in P$ to $q(p) \in \Re$. *q(p)* is to be thought of as the strength of the propagated input at place *p*.
- $s : P \to \Re$ is a total mapping that maps a place $p \in P$ to $s(p) \in \Re$. *s(p)* is to be thought of as the strength of the pheromone at place *p*.

If a scenario does not specify the initial state of the *PI*, $q(0)=0$ and $s(0)=0$ is assumed by default.

The *PI* emulates the evaporation and the propagation of pheromones in the real world. Evaporation decreases the local pheromone strength, whereas propagation spreads the effect of an input to the *PI* to the surrounding neighborhood of the place at which an external input occurred. The rate with which a pheromone evaporates is specified in the evaporation parameter of the pheromone. The reduction in strength of the input in a propagation step is influenced by the propagation parameter of the pheromone. All

pheromones of the same type must share the same evaporation and the same propagation parameter.

**Definition 3 (Parameters)**

The real values $E$ and $F$ are two *parameters* of the pheromone. $E$ ($E \in (0,1)$) is the evaporation parameter and $F$ ($F \in [0,1)$) is the propagation parameter of the pheromone.

In a given scenario, the agents located at places of an environment jointly generate an external spatial-temporal input pattern to the pheromone. For every moment and for each place the input pattern specifies the strength of the external input. From the perspective of the *PI*, such a pattern completely specifies the scenario. To guarantee the global stability of the *PI* (Section 3.2.2), the external input pattern is globally limited in strength.

**Definition 4 (External Input Pattern)**

Let <*P,N*> be an environment. The *external input pattern* in <*P,N*> is given by the real number $R$ and the mapping $r$, where

- $R \in \Re$ is the global limit of the external input pattern, and
- $r : \aleph \times P \to (-R, R)$ is a total mapping that maps every point in time $t \in \aleph$ and each place $p \in P$ to $r(t, p) \in (-R, R)$. *r(t,p)* is to be thought of as the strength of the external input at place $p$ and at time $t$.

In the analysis of scenarios, external spatio-temporal input patterns are often considered separately for each place. In this case the following notation of an input sequence at a place is used.

**Notation 2 (External Input Sequence)**

An external input pattern considered for only one place $p \in P$ is called the *external input sequence* at $p$. Let $R$ and $r'$ represent an external input pattern. The mapping $r : \aleph \to (-R, R)$ that maps a point in time $t \in \aleph$ to $r(t) \in (-R, R)$ is the input sequence at $p$ if *r(t)=r′(t,p)* holds for all $t$.

The state of the *PI* changes over time through external input, internal propagation, and continuous evaporation. The following transition functions formally specify the evolution of the state. The first transition function specifies the propagation of input events through the *PI* and the second function combines the input from the agents and the propagated input with a general dispersion function to determine the current strength of the pheromone.

**Definition 5 (Transition Functions)**

Let <*P,N*> be an environment, let *E* and *F* be the parameters of the pheromone, and let *R* and *r* be an external input pattern. The change of the state of a *PI* is determined by the two *transition functions* *q* and *s*, where

- $q : \aleph \times P \to \Re$ is a total mapping that maps a point in time $t \in \aleph$ and a place $p \in P$ to $q(t, p) \in \Re$ and
- $s : \aleph \times P \to \Re$ is a total mapping that maps a point in time $t \in \aleph$ and a place $p \in P$ to $s(t, p) \in \Re$.

Specifically, the transition of the propagated input pattern (*q*) is defined as

$$q(t+1, p) = \sum_{p \in N(p')} \frac{F}{|N(p')|} \left( r(t, p') + q(t, p') \right) \tag{3.1}$$

And the transition of the pheromone strength pattern (*s*) is defined as

$$s(t+1, p) = s(t, p) * E + r(t, p) + q(t, p) \tag{3.2}$$

Besides the external input generated by local agents, input events propagated from neighboring places change the strength of the pheromone. The mechanism of a "wave"-like propagation of an input event through the *PI* is captured in the definition of the transition function *q* (3.1). Every transition takes all input events at each place and propagates them to the direct neighbors according to the propagation relation.

Multiplied with the propagation parameter *F*, there is a reduction in the absolute strength in every propagation step. $|N(p')|$ influences the strength of an input event when it is propagated from place $p' \in P$ to place $p \in P$. The term represents the number of neighbors of *p'* according to the propagation relation *N*. The *N(p')* selected in a transition are never empty, because *p* is one of these neighbors.

The strength of inputs propagated from different places to *p* add up in every step. The strength of an input propagated on from *p'* to the neighbors of *p'* (*N(p')*) is split among these neighbors. Thereby, auto-reinforcement effects in cyclic propagation are prevented (Section 3.2.2).

The local strength (*s(p)*) models a quantity of the pheromone. Negative values for the pheromone strength are permitted too, intuitively representing an amount of "anti-pheromone". But, only the context of an application specifies the exact semantics of the pheromone.

The model chosen for the *PI* has three basic features. It is finite and discrete in space (set of places *P*), infinite and discrete in time ($t \in \aleph$), and the model is infinite and continuous in its states ($q \in \Re$, $s \in \Re$).

In most cases a scenario-based analysis may consider one pheromone at a time. Only when considering local strength patterns at one place, multiple pheromones enter the analysis. The topological structure, the pheromones with their propagation direction and their evaporation and propagation parameters, and the input behavior of the agents have to be formally modeled to analyze how the state of the *PI* evolves in a specific scenario.

**Notation 3 (Scenario)**

A (one-pheromone) *scenario* $S=<L,E,F,R,r>$ specifies an environment $L=<P,N>$, pheromone parameters $E$ and $F$, and an external input pattern ($R$, $r$). Optionally, the initial state of the *PI* ($q(0)$, $s(0)$) may be specified by the scenario.

### 3.2.2 Proof of Stability

The Global Stability theorem is stated for an arbitrary scenario. The first stage of the proof introduces a global majorant for the external input, knowing that the desired stability is achieved when the *PI* is stable in face of the majorant input pattern already. In a second step, the local dampening of the aggregated input at a set of places is shown.

Finally, all ingredients are assembled to take the main step, showing that there is an upper limit to the aggregated effect of one external input at one place on the pheromone strength at any place. The additivity of the transition function (3.2) is used to extend the argument to infinite input sequences at all places.

**Theorem 1 (Global Stability)**

Let $S=<<P,N>,E,F,R,r>$ be an arbitrary scenario. There exists a fixed upper limit $B$ ($B \in \Re^+$) for the absolute value of the pheromone strength at any place in $p \in P$ ($|s(p)|$) in any state of the *PI*.

Without loss of generality it is assumed in the following that all external input values are either equal to or larger than zero.

An external input $r(t,p)$ at a place $p \in P$ and at time $t \in \aleph$ adds to the local pheromone strength. A constant input pattern $r' \equiv R$, adding $R$ at each place and at every moment, is the majorant for the input pattern $r$. If the *PI* remains stable faced with an input $r'$, global stability is also guaranteed for the input $r$.

#### Local Stability

Consider an arbitrary set of places and an aggregated input (the sum of external and propagated input) to it at one point in time. Local stability of the *PI* guarantees an output of weaker strength propagated to the neighbors of the chosen set in the next step. The truth of the statement becomes clear with the following additional definition that is only required for the proof of stability.

**Definition 6 (Input/Output)**

Let *S=<<P,N>,E,F,R,r>* be an arbitrary scenario.

An <u>*aggregated input at a set of places*</u> is a mapping $I:\aleph\times P^n\to\Re$ that maps a point in time $t\in\aleph$ and any subset of $P$ ($P'\subseteq P$) to $I(t,P')\in\Re$. Following the definition 5, an aggregated input at a set of places is given by:

$$I(t,P')=\sum_{p\in P'}\left(r(t,p)+q(t,p)\right) \tag{3.3}$$

A <u>*single output out of one place into another one*</u> is a mapping $o:\aleph\times P\times P\to\Re$ that maps a point in time $t\in\aleph$ and two arbitrary places $p,p'\in P$ to $o(t,p,p')\in\Re$. Following the definition 5, a single output is given by:

$$o(t,p,p')=\begin{cases}\dfrac{F}{|N(p)|}\left(r(t,p)+q(t,p)\right) & : p'\in N(p)\\ 0 & : p'\notin N(p)\end{cases} \tag{3.4}$$

An <u>*aggregated output out of a set of places*</u> is a mapping $O:\aleph\times P^n\to\Re$ that maps a point in time $t\in\aleph$ and any set of places $P'\subseteq P$ to $O(t,P')\in\Re$. Following the definition 5, an aggregated output out of a set of places is given by:

$$O(t,P')=\sum_{p\in P'}\sum_{p'\in N(p)}o(t,p,p') \tag{3.5}$$

*O(t,P')=F*I(t-1,P')* holds at any point in time and for any subset of places. Therefore, the local weakening of an input required for local stability is given, because *F<1* holds.

**Propagation Stability**

Local stability does not provide an approximation of the limit of all strength values. In the following the aggregated effect of one external input is approximated for an arbitrary place. But, to formally state the propagation stability theorem, a one-time and one-place external input pattern *r''* must be declared first.

Assume that at the arbitrary place $p_{in}\in P$ one and only one external input of strength *1* at time *t=0* is generated. No other external input is ever observed at any place. The external input is formally specified as:

$$r''(t,p)=\begin{cases}1 & : p=p_{in}\wedge t=0\\ 0 & : p\neq p_{in}\vee t\neq 0\end{cases},\ p\in P,t\in\aleph \tag{3.6}$$

**Theorem 2 (Propagation Stability)**

There exists a fixed upper limit to the aggregated sum of all propagated inputs at an arbitrary place if a one-time and one-place external input is assumed.

In the following, sets of places are considered and hence, additional notations have to be declared first.

**Notation 4 (Notations for Sets of Places)**

Let <*P,N*> be an environment.

The *neighbors of a set of places* $P' \subseteq P$ according to the propagation relation *N* are denoted by $N(P') = \{p : \exists p'(p' \in P \wedge p \in N(p'))\}$.

The *number of neighbors of a set of places* $P' \subseteq P$ according to the propagation relation *N* is denoted by $Z(P') = \sum_{p' \in P'} |N(p')|$.

The *number of occurrences of a place* $p \in P$ in the neighborhood of a set of places $P' \subseteq P$ according to the propagation relation *N* is denoted by $X(p, P') = |\{p' : p' \in P' \wedge p \in N(p')\}|$.

The following statement always holds true: $X(p, P') \leq Z(P')$. The statement is central to the following approximation of the upper limit of the local pheromone strength.

*Proof (Theorem 2).* Consider an arbitrary place $p_x \in P$. The proof of theorem 2 aggregates the propagated input at $p_x$ while it follows the propagation of the effects of the single external input.

Let the set $P_0 = \{ p_{in} \}$ comprise only the place where the one external input occurred. Furthermore let $P_i$ denote the set of neighbors of $P_0$ that are reached in the *i*-th propagation step given by $P_i = N(P_{i-1})$ $(i>0)$. According to the previous discussion of local stability and assuming the specific one-time and one-place external input pattern, the aggregated input at the places of $P_i$ at the *i*-th propagation step is given by $I(i+1,P_i)=F^i$ and it occurs at time $t=i+1$.

Assume that an arbitrary step in the propagation takes the input events from set $P_{i-1}$ to set $P_i$. The overall input into $P_i$ is $I(i+1,P_i)=F^i$. A portion of the input, specifically $F^i * \frac{X(p_x, P_{i-1})}{Z(P_{i-1})}$, is propagated to $p_x$. Therefore, the propagated input to $p_x$ aggregated for all time is given by:

$$\sum_{t=1}^{\infty} q(t, p_x) = \sum_{t=1}^{\infty} F^t * \frac{X(p_x, P_{t-1})}{Z(P_{t-1})} \tag{3.7}$$

With $\frac{X(p_x, P_{t-1})}{Z(P_{t-1})} \leq 1$ holding, the infinite sum in (3.7) is always limited by:

$$\sum_{t=1}^{\infty} F^t = \frac{F}{1-F} \tag{3.8}$$

The sum of the effects of propagated inputs resulting from one external input of strength *R* at an arbitrary place is always finite and limited by $R * \frac{F}{1-F}$.

**Conclusion**

After the effect of one external input at one place on all other places is proven to be

limited, the majorant input pattern $r' \equiv R$ is considered.

One external input of strength $R$ to one place increases the strength at each place by maximally $R + R * \frac{F}{1-F} = R * \frac{1}{1-F}$. With each place receiving an external input of $R$ at every unit time, an aggregated input of $R * |P| * \frac{F}{1-F}$ at any time and at each place results. In the case of such a fixed aggregated input, at any moment $t$ the pheromone strength at each place $p \in P$ is smaller than:

$$s(0,p) * E^t + R * |P| * \frac{1}{1-F} * \sum_{i=0}^{t-1} E^i \tag{3.9}$$

The given limit may be approximated for large $t$ by:

$$B = R * |P| * \frac{1}{1-F} * \frac{1}{1-E} \tag{3.10}$$

The pheromone strength at all places in the *PI* remains finite under the majorant external input $r' \equiv R$. Hence, the infrastructure remains stable under the original input $r$. Thereby the Global Stability theorem (Theorem 1) is proven.

### 3.2.3 Single-Pheromone Analysis

Most features of the *PI* are discussed for one pheromone alone. In the following, $E$ and $F$ represent the evaporation and propagation parameter of an arbitrary pheromone in the respective scenario. Furthermore, for every external input pattern an arbitrary global limit is assumed without always stating it explicitly.

**Single-Place Scenarios**

Let $<P_s,N_s>$ be an environment. The tuple represents a *single-place environment*, if $P_s=\{ p_s \}$ holds. $N_s(p_s) = \varnothing$ results from the irreflexivity of $N_s$. In the following, $p_s$ denotes the single place in the environment of the scenarios. For one place the transition function (3.2) may be reduced to:

$$s(t+1) = s(t) * E + r(t), t \in \aleph \tag{3.11}$$

The no-input scenario with $r \equiv 0$ is the simplest one to consider. There, the transition function is further reduced to $s(t+1)=s(t)*E=s(0)*E^{t+1}$. The initial strength $s(0)$ simply evaporates over time.

An external input is taken into account in all other scenarios. The following analysis considers the transition function (3.11) in its non-recursive form:

$$s(t) = s(0) * E^t + \sum_{i=0}^{t-1} \left( r((t-1)-i) * E^i \right), t > 0 \tag{3.12}$$

A basic scenario to consider is a repeated constant input by one agent located at place $p_s$. The aim of the agent is to reinforce the relevance of information expressed by the pheromone. The behavior is called refreshing the pheromone. Hence, the input scenario is

a one-agent regular-refresh scenario.

***One-Agent Regular-Refresh Scenario***

For the analysis of the pheromone strength at $p_s$, it is sufficient to specify the input strength $A \in \Re$ and the input rate $T \in \aleph^+$. The actual motivation of the input behavior is irrelevant. Without loss of generality it is assumed throughout the following discussion that the refresh strength $A$ is larger than zero.

Knowing $A$ and $T$, $r$ is formally expressed by

$$r(t) = \begin{cases} A & : \exists j (j \in N \wedge t = Tj) \\ 0 & : \text{all else} \end{cases}$$

Based on the regular input sequence, the transition function is reduced to

$$s(t) = s(0) * E^t + A * \sum_{i=0}^{j} E^{(t-1)-Ti} \tag{3.13}$$

with $j$, the number of previous inputs, specified as: $j \in \aleph \wedge Tj < t \wedge !\exists k (k \in \aleph \wedge k > j \wedge Tk < t)$.

Are there one or more fixed points – equilibria – in the evolution of the strength at $p_s$? The location of fixed points in a scenario is very important because they determine the most probable results when sampling the pheromone strength. If there exists a close approximation of a fixed point, then agent scenarios may be designed and tuned analytically.

In the one-agent regular-refresh scenario equilibrium is reached. The following analysis formally describes its location, the current distance to the equilibrium, and the time it takes to reach it.

***Equilibrium***

The transition function *s(t)* is considered for large $t$ to derive the equilibrium the pheromone strength reaches. The function does not evolve continually. It has a local maximum at $t_m = Tn + 1, n \in \aleph$ followed by a steady decline until it reaches its local minimum at $t_m+T-1=T(n+1)$. Then, it jumps up to the next maximum at $t_m+T=T(n+1)+1$ (Figure 3.1).

For later discussion it must be noted that the local minimum at $t_m+T-1$ is only the "visible" minimum in the discrete-time model. When considering the scenario in continuous time, the evaporation continues right to the moment, the input strength is added. There, at a moment of discontinuity, the local minimum and maximum meet. The local maximum "hides" the actual minimum in the discrete model.

The local maximum at time $t_m=Tn+1$ is given by:

$$s(t_m) = s(0) * E^{Tj+1} + A * \sum_{i=0}^{j} E^{Ti} \tag{3.14}$$

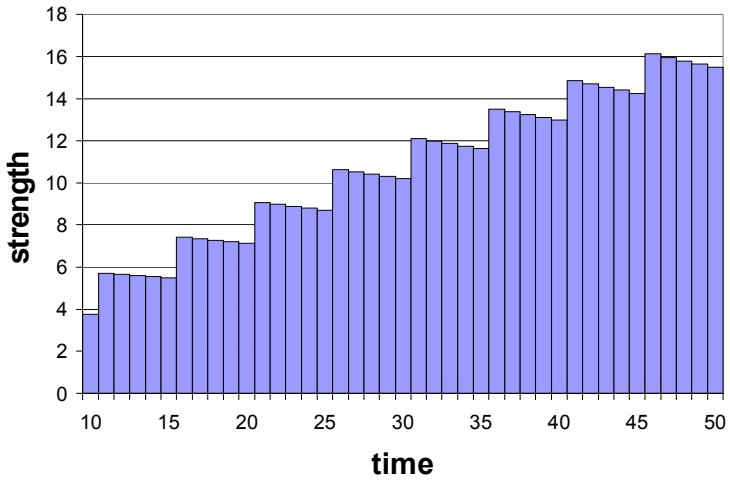


Figure 3.1. Pheromone Strength under Regular Refresh – A=2, T=5, E=0.99

The sequence of local maxima has a fixed point, denoted by $B_m$. It is the limit value of equation (3.14), given by:

$$B_m = \frac{A}{1 - E^T} \tag{3.15}$$

With $B_m$ approximating the local maxima for large $t$, $B_m$-$A$ or $B_m$*$E^T$ gives the "hidden" local minima. Both terms are valid. The first one follows the reasoning that the pheromone is refreshed regularly by an input of $A$, whereas the second term represents the evaporation of the pheromone strength throughout the time between two refresh moments. $B_m$*$E^{T-1}$ gives the visible local minima.

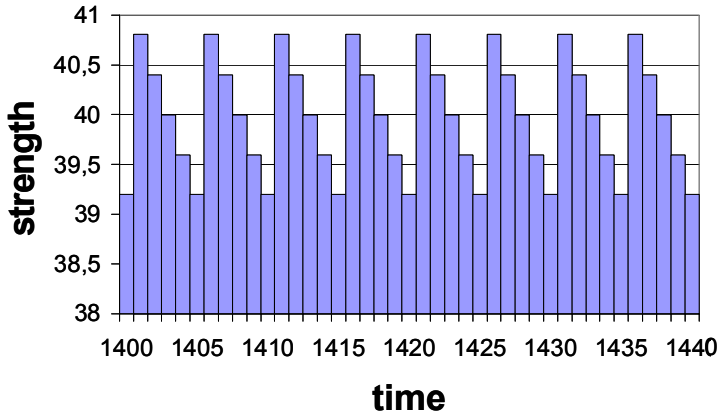


Figure 3.2. Pheromone Strength under Regular Refresh – Equilibrium

Hence, the range that limits the pheromone strength at equilibrium is

$$[B_m, B_m - A] = \left[\frac{A}{1 - E^T}, A\left(\frac{1}{1 - E^T} - 1\right)\right] \tag{3.16}$$

Figure 3.2 illustrates the equilibrium state reached under the regular refresh behavior of $A=2$, $T=5$. The evaporation parameter is set to $E=0.99$.

***Distance to Equilibrium***

With an analytic description of the equilibrium range at hand, the next step in the analysis focuses on the current distance to the equilibrium. Since the equilibrium is specified by the local minimum and maximum of the strength, the distance between a local maximum and the upper boundary of the equilibrium range is considered.

Let $d_m(t_m)$ denote the distance to the equilibrium at a maximum moment $t_m$. $d_m(t_m)$ is given by

$$d_m(t_m) = \left|B_m - s(t_m)\right| = \left|B_m * E^T - s(0) * E\right| * E^{Tj}, t_m = Tj + 1, j \in \aleph \tag{3.17}$$

Under what conditions does $d_m(t_m)=0$ hold? Whereas $T = \infty$ as a solution for $d_m(t_m)=0$ does not tell much, $Tj \to \infty$ is also fulfilled with $t_m \to \infty$ since $Tj=t_m-1$. Thus, the equilibrium is approached by the pheromone strength, but it is reached after infinite time only.

Another solution to $d_m(t_m)=0$ is $s(0)=B_m*E^{T-1}$, the "visible" local minimum. The first transition step in the formal model reduces the initial value to $B_m*E^T$ – the "hidden" minimum – and simultaneously adds the input $A$. Thus, $s(1)=B_m*E^T+A=B_m$ is the local maximum of the equilibrium state.

In real (continuous) time, the initial strength would have been set to $B_m$ to start right in the equilibrium state. This difference between the model and reality vanishes only for $T=1$. In this specific case, there is a refresh at every moment and no local minima result in the discrete model. In such a scenario the analytically predicted equilibrium collapses into one point

$$B_m = A * \frac{1}{1-E} \tag{3.18}$$

and the expression of the distance to the equilibrium is reduced to

$$d_m(t_m) = \left|B_m - s(0)\right| * E^{t_m} \tag{3.19}$$

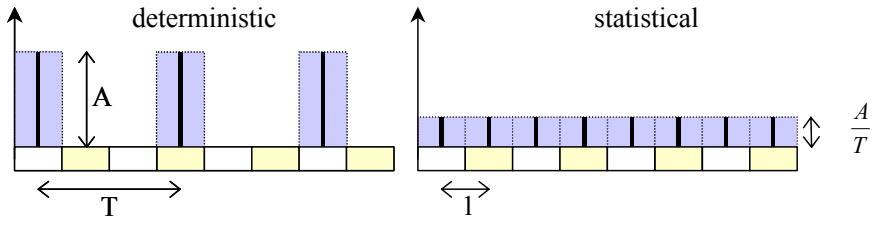


Figure 3.3. The Deterministic and the Statistical View on a Refresh Scenario

If the model is most accurate for $T=1$, how may other scenarios be translated into such a configuration? In the previous discussion, the one-agent regular-refresh scenario was considered in a deterministic way only. It is completely determined when the pheromone is refreshed by an input of strength $A$. The event happens **every** $T$ **time steps**. A less discrete stance would consider a refresh by $A$ **over a period of** $T$, statistically resulting in an average refresh by $A/T$ at every unit time (Figure 3.3). In this case, only one fixed point exists for all sub-sequences of the strength at $p_s$. It is denoted $B$ and given by:

$$B_m = \frac{A}{T} * \frac{1}{1-E} \tag{3.20}$$

A less deterministic view on a given scenario may reduce the complexity of the analytic expressions gained and even remedy differences that are introduced in the model's abstraction from reality. But, with only one agent in the scenario and with its refresh rate restricted to natural numbers, the strict deterministic view is more correct when considering the long-term evolution of the pheromone strength. The strength value oscillates widely between the local minima and maxima for larger refresh strength values $A$ or longer refresh rates $T$.

In the deterministic view, the equilibrium is given by the local extrema, whereas in the statistical stance an average pheromone strength is predicted. Figure 3.4 illustrates the difference.

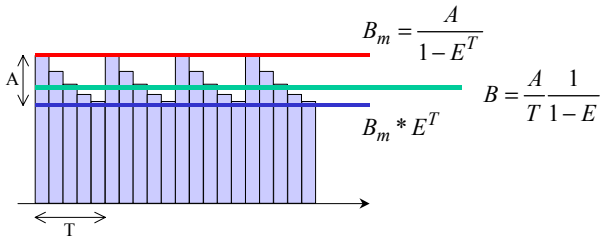


Figure 3.4. Location of the Predicted Equilibrium

The remaining analysis of the one-agent regular-refresh scenario considers both perspectives, whereas the statistical view is more appropriate to handle multi-agent scenarios.

### *Time to Equilibrium*

Generally, the equilibrium state is only approximated in finite time. But, in the process of tuning pheromone-based coordination mechanisms, it is sufficient to know when the approximation of the equilibrium state has reached a certain quality. Assume the following quality measure: The approximation is sufficiently close if the distance between the current and the equilibrium value is smaller than $x$ percent ($x \in (0,1)$) of the input strength. The criterion is formally stated as $d_m(t_m) < |\, x * A \,|$. The time to reach the equilibrium state when applying the quality measure is given by

$$t_m \geq \frac{1}{\ln(E)} \ln\left(\left|\frac{x * A}{B * E^T - s(0) * E}\right|\right) \tag{3.21}$$

In the statistical stance, the criterion is stated as $d(t) < |x*A/T|$ and the time $t$ fulfilling it is given by:

$$t > \frac{1}{\ln(E)} \ln\left(\left|\frac{x * A / T}{B - s(0)}\right|\right) \tag{3.22}$$

### *Sampling under Regular Refresh*

Assume that the equilibrium for a pheromone under a specific refresh scenario is predictable. The predictability may be used to explicitly communicate data from one agent

to another based on the observed strength.

Consider again the previous scenario where one agent refreshes a pheromone regularly. In the following the agent is called “Sender”. Another agent (“Receiver”) is added to the scenario that samples the current strength of the pheromone. The following discussion presents approximations that enable Receiver to derive the parameters specifying the refresh behavior of Sender. These parameters may carry numerical data from Sender to Receiver.

Let $S$ denote the pheromone strength sampled at an arbitrary moment. The primary assumption on the side of the Receiver is that the pheromone strength has reached equilibrium already. Taking the deterministic stance, $S$ must be within the equilibrium range of $[B_m\text{-}A,B_m]$. In the statistical view, $S$ is equal to the fixed point $B$.

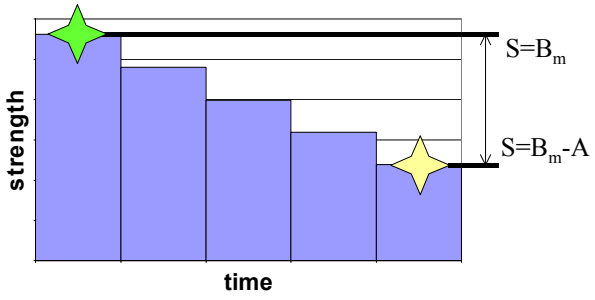


Figure 3.5. Sampling the Equilibrium Range

The deterministic view is considered first, discussing the two extrema $S=B_m\text{-}A$ and $S=B_m$ (Figure 3.5). In the first case, the local minimum was sampled. Formally, the parameters of Sender, as perceived by Receiver, are given by $A_{min}=S(1\text{-}E^T)E^{-T}$ and $T_{min}=ln(E)^{-1}ln(S/(S+A))$ respectively. At the other extreme, the parameters are computed by $A_{max}=S(1\text{-}E^T)$ and $T_{max}=ln(E)^{-1}ln((S\text{-}A)/S)$ respectively. Consequentially, Receiver has to know one refresh parameter to deduce the other.

Based on the discussion of these extreme cases, Receiver has an interval to choose from when deducing the missing parameter of Sender. If it knows the refresh strength $A$, then it may pick one value from $[T_{min},T_{max}]$ for the perceived refresh rate. Or, it knows the refresh rate $T$. Then, it is the interval $[A_{min},A_{max}]$ from which to select a refresh strength value.

Assume that Receiver picks the perceived parameter from the center of the available range. In this case, it may directly use the following equations:

$$A_D = \frac{S}{2E^T}(1-E^T)(1+E^T) \text{ and } T_D = \frac{1}{2\ln(E)}\ln\left(\frac{S-A}{S+A}\right) \tag{3.23}$$

Having sampled the pheromone strength only once, Receiver does not know the actual position of S in the interval $[B_m\text{-}A,B_m]$. Thus, there may be an error in its deduction using equations (3.23). The maximum error is given by:

$$A_D^{err} = \left|\frac{S}{2E^T}(1-E^T)^2\right| \text{ and } T_D^{err} = \left|\frac{1}{2\ln(E)}\ln\left(\frac{S^2-A^2}{S^2}\right)\right| \tag{3.24}$$

The statistical view states that the sampled value $S$ is equal to the equilibrium value $B$. Thus, the parameters are simply computed as:

$$A_S = S * T * (1 - E) \text{ and } T_S = \frac{A}{S}\frac{1}{1-E} \tag{3.25}$$

With every sampled value assumed to be the equilibrium value, there is no error in the deduction of the parameters.

***Multi-Agent Scenarios***

The previous scenario of one agent regularly refreshing the pheromone at $p_s$ is now extended to $n$ agents showing the same behavior.

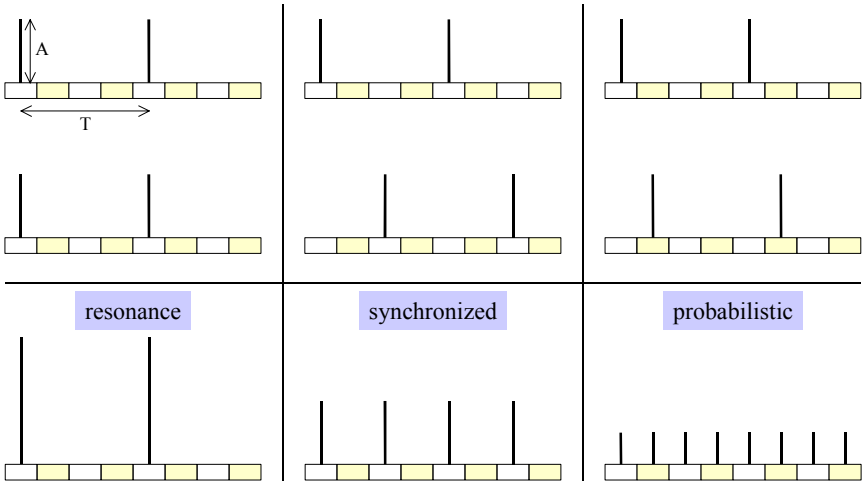


Figure 3.6. Approaches to Scenario Translation

With more than one agent at $p_s$ regularly refreshing the pheromone, the input pattern ($r$) changes. There are three different approaches available to describe the new pattern in terms of the already analyzed one-agent scenario. The refresh behavior in the one-agent scenario is denoted by $A_1$ and $T_1$. It approximates the multi-agent refresh behavior where each agent refreshes the pheromone by $A$ in a rate of $T$ (Figure 3.6).

Table 3.1. Translation Approaches in the Multi-Agent Scenario

| translation approach | refresh strength ($A_1$) | refresh rate ($T_1$) |
|---|---|---|
| deterministic resonance | $n*A$ | $T$ |
| deterministic synchronized | $A$ | T/n |
| probabilistic | $n*A/T$ | $1$ |

The *deterministic resonance approach* assumes the input from all agents to arrive at the same moment. Thus, the refresh strength of all inputs adds up. The refresh rate remains the same indifferent of the number of agents ($A_1=n*A$, $T_1=T$).

The *deterministic synchronized approach* assumes the refresh actions of all agents to occur evenly spaced over time. Thus the refresh strength of every aggregated input is equal to the input from one agent. The refresh rate, on the other hand, depends on the number of agents. In the discrete-time model such an approach does not seem appropriate, since it restricts $n$ to those numbers where $T / n \in \aleph$ holds ($A_1=A$, $T_1=T/n$).

A third approach is the *probabilistic approach*, which is strongly related to the statistical view. It considers the probability for a refresh action of one agent to happen at a specific moment in an interval of *T*. The probability is *1/T*. Hence, the probable refresh strength for every moment in an interval of *T* is *A/T* with one agent, as argued in the statistical stance. With *n* agents, the refresh strength increases to $A_1=n*A/T$ with a refresh rate of $T_1=1$.

As in the analysis of the one-agent regular-refresh scenario, the probabilistic approach lacks the inconveniences dictated by the choice of the formal model. Hence, the probabilistic translation of multi-agent scenarios is selected for the following analysis.

In the multi-agent regular-refresh scenario the equilibrium is the most important feature to discuss too. It is determined translating equation (3.15) on the basis of the probabilistic approach. The equilibrium value for *n* agents is denoted by *B(n)*:

$$B(n) = n\frac{A}{T}\frac{1}{1-E} \tag{3.26}$$

As the equation indicates, there is a convenient linear relation between the number of agents present and the equilibrium value of the pheromone strength.

*Distance between two Equilibria*

Let *D(n,m)* denote the distance between the equilibrium reached with *n* agents and the one with *m* agents (Figure 3.7). It is given by:

$$D(n,m) = \left|B(m) - B(n)\right| = \left|(m-n)\frac{A}{T}\right|\frac{1}{1-E} \tag{3.27}$$

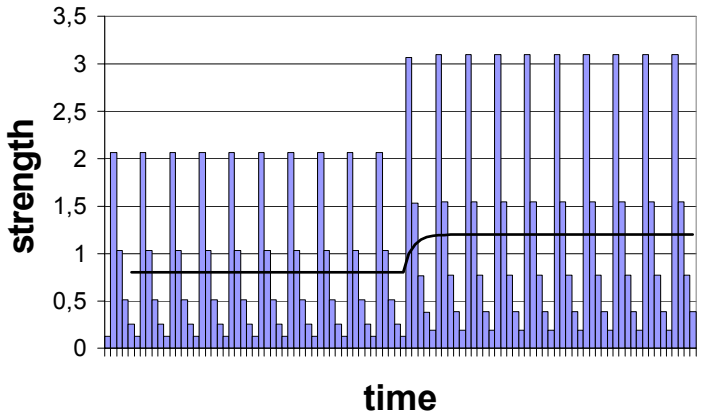


Figure 3.7. Changing the Number of Agents

The prediction of the distance between two equilibria may be used to tune the quality of perception of an agent according to the respective requirements of the designed interaction mechanism.

*Time between two Equilibria*

Assume that the number of agents refreshing the pheromone changes over time. In this case, it is important to know how long it takes to come sufficiently close (by a *x*-percent threshold) to the new equilibrium. The responsiveness of agent interactions built upon the

*PI* depends on the time it takes for the pheromones to stabilize at the new equilibrium. Similarly to the one-agent case (statistical view) the "sufficiently close" criterion for a change from *n* to *m* is expressed in the statement:

$$d(t) < \left| x * m \frac{A}{T} \right| \tag{3.28}$$

The new equilibrium *B(m)* is approximated. The values of *t* fulfilling the statement (3.28) are given by

$$t \geq \frac{1}{\ln(E)} \ln\left( \left| \frac{m}{m-n} \right| * x(1-E) \right) \tag{3.29}$$

It should be noted that the required time neither depends on the refresh strength of an agent, nor does it depend on its refresh rate.

***Providing Regularity in a Different Way***

The scenario of multiple stationary agents is easily translated to a multi-agent scenario with mobile agents, if the input pattern is analyzed instead of the agent behavior creating it. Consider a flow of mobile agents passing sequentially through $p_s$. Each agent in the flow generates one input and then it leaves (Figure 3.8).

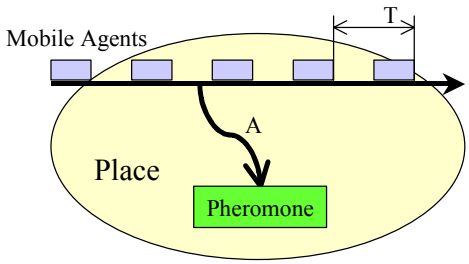


Figure 3.8. Multi-Agent Mobility Scenario

The number of agents $n \in \aleph^+$ passing through $p_s$ in a unit time characterizes the flow. *n* is also called the "load" of $p_s$. The input behavior of an agent is specified by the strength $A \in \Re^+$ of the single input. As a consequence, there is an average input of *n*A* strength every unit time, taking the pheromone strength at $p_s$ to a fixed point of:

$$B(n) = n * A * \frac{1}{1-E} \tag{3.30}$$

A change in the strength of the flow from *n* agents to *m* agents ($m \in \aleph^+$) results in a change in the location of the fixed point. The distance of two equilibria and the time to change to the new equilibrium is computed as in the stationary agent scenario (Equations (3.27), (3.29)).

*Visibility of Load Changes*

Assume that the number of agents passing through place $p_s$ changes from *n* to $n+\delta, \delta \in \Re^+$ agents per unit time. The change is considered visible in the pheromone strength if the distance between the respective equilibria is sufficiently large. As in the discussion of the time to equilibrium in the context of a single agent scenario, a specific

percentage ( $x \in [0,1]$ ) of the input strength is used to define the criterion:

$$D(n, n+\delta) > \left| x * A * (n+\delta) \right| \tag{3.31}$$

Thus, to become visible, the strength of the change of the load at $p_s$ must fulfill:

$$\delta > n \frac{x(1-E)}{1-x(1-E)} \tag{3.32}$$

*Visibility of Perturbations*

In the last step of the analysis of the mobile-agent scenario, the duration of the change of the load at $p_s$ is restricted in time. Whereas equation (3.32) considers the visibility of a permanent change, during a temporary change – a perturbation – the pheromone strength may not reach equilibrium.

A perturbation in the load of $p_s$ is formally characterized by the following parameters. Assume that a flow of $n$ ($n \in \Re^+$) agents per unit time passes through $p_s$. The duration of the perturbation is specified by $\theta$ ($\theta \in \aleph^+$), while $\delta$ ($\delta \in \Re^+$) specifies the strength of the perturbation. Finally, $t_x$ ($t_x >> 0$) is the moment the perturbation starts. The load changes back to $n$ at $t_x + \theta$. Figure 3.9 illustrates the effect of a perturbation as seen in the strength of the refreshed pheromone.

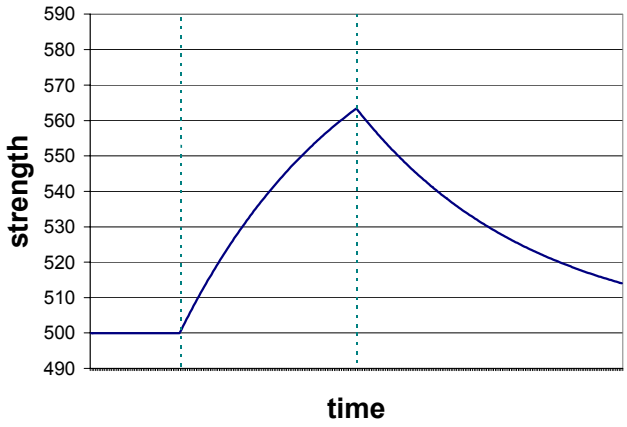


Figure 3.9. Effect of a Perturbation

The fixed point under the new load may not be reached during $\theta$. Instead, the strength right at the end of the perturbation has to be considered for the restriction of the visibility. Thus the visibility criterion is stated as

$$\left| s(t_x + \theta) - B(n) \right| > \left| x * A * (n+\delta) \right| \tag{3.33}$$

Assume a fixed duration of a perturbation $\theta$. According to the criterion, the perturbation is visible in the pheromone strength if the strength of the change fulfills the following condition:

$$\delta > n * \left| \frac{E^{\theta} - 1}{1 - E} * \frac{1}{x - \sum_{i=0}^{\theta-1} E^{i}} - 1 \right| \tag{3.34}$$

The minimum duration $\theta$ for visible perturbations of a fixed strength $\delta$ may be predicted similarly.

**Multi-Place Environments**

After considering the pheromone strength at the single place $p_s$, the following discussion extends the scope of the analysis to the effects of spatial propagation of input events.

The following two scenarios specify different environments. First a simple sequence of places is assumed. In a second environment, the ends of the sequence are joined to form a circle to explore the effects of cyclic propagation.

***Sequential Topologies***

Let $<P_{seq},N_{seq}>$ be an environment. It represents a sequential topology of $n$ places ($n \in \aleph$, $n>1$) if the following two statements hold:

- $P_{seq} = \{ p_i: i=1..n \}$, and
- $\forall i(i \in [2,n] \rightarrow N_{seq}(p_{i-1}) = \{p_i\}) \wedge N_{seq}(p_n) = \varnothing$.

In every propagation step, the number of neighboring places is never larger than one. Therefore, in equation (3.1) the term "$|N(p')|$" may be replaced by a constant value of one. In the following discussion, the term $s_i(t)$ denotes the pheromone strength at place $p_i$ at time $t$.

One stationary agent provides the first input scenario. Without loss of generality, the agent is located at place $p_1$ and it expresses a regular-refresh behavior specified by the parameters $A$ (refresh strength) and $T$ (refresh rate). The agent provides the only external input.

Because of the sequential nature of the topology, the strength at $p_1$ is only changed by the external input from the agent. No additional strength comes through spatial propagation. As far as $s_1$ is concerned, the situation is the same as in the previous single-place one-agent regular-refresh scenario. $B_i$ denotes the fixed point in the sequence of the strength values at the place $p_i$. For $p_1$ the fixed point is given by $B_1 = A/T/(1-E)$.

Consider $s_2(t)$, the pheromone strength at the next place in the sequence. There, the refresh by the agent at $p_1$ arrives through one step of spatial propagation weakened by the multiplication with $F$, the propagation parameter (Figure 3.10).

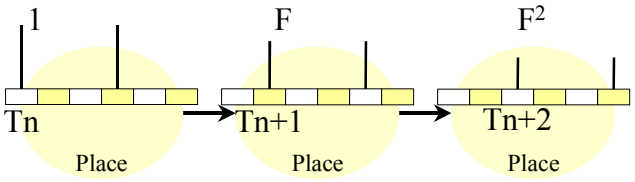


Figure 3.10. Diminished Effects of Distant Refreshes

Extending the argument to the other places in the sequence, the equilibrium at an arbitrary place in the sequence is predicted as:

$$B_i = F^{(i-1)} \frac{A}{T} \frac{1}{1-E}, i = 1..n \tag{3.35}$$

Exemplary for multi-agent scenarios in a sequential topology, a two-agent scenario is considered. The first agent resides at place $p_1$, whereas a second one is located a different place $p_x$, $x \neq 1$. Both agents exhibit the regular-refresh behavior specified by the parameter $A$ and $T$.

The strength at the places $\{ p_1, ..., p_{x-1} \}$ evolves towards the equilibrium given by equation (3.35). The remaining places $\{ p_x, ..., p_n \}$ also receive input from the actions of the second agent.

At place $p_x$, the external input from the second agent directly affects the pheromone. Additionally, it receives the propagated input from the first agent reduced to $F^{x-1}*A/T$. Hence, the equilibrium at $p_x$ is given by $(1+F^{x-1})\frac{A}{T}\frac{1}{1-E}$. The prediction of the equilibrium at an arbitrary place is:

$$B_i = F^{(i-x)}(1+F^{(x-1)})\frac{A}{T}\frac{1}{1-E}, i = x..n \tag{3.36}$$

In the same way any other multi-agent scenarios in a sequential topology may be analyzed. There remains no restriction on the position of the agents or on their respective regular-refresh behavior.

### *Cyclic Topology*

With the analysis of sequential topologies completed, the next step is to analyze topologies with loops. Within such loops, spatial propagation of one input event is repeated infinitely.

Let $<P_{cyc},N_{cyc}>$ be an environment. The environment represents a single loop of $n$ places ($n \in \aleph$, $n>1$) if the following two statements hold:

- $P_{cyc} = \{ p_i: i=1..n \}$, and
- $\forall i(i \in [2,n] \rightarrow N_{cyc}(p_{i-1}) = \{p_{i-1}\}) \wedge N_{cyc}(p_n) = \{p_1\}$.

The pheromone strength at a place $p_i \in P_{cyc}$ is denoted by $s_i$, and as in sequential environments pheromone strength is not split during propagation to neighbors in a single-loop layout.

Consider one agent located at place $p_x$. With all places being part of the loop it does not make a difference where the agent resides. As in most of the previous scenarios, the agent

exhibits a regular-refresh behavior of an average strength *A/T*. Following the definition of the spatial propagation of an input event in the formal model, it takes *n* units time for an input event to be propagated around the *n*-places loop. During a one-loop propagation the refresh strength is reduced to $F^n * A/T$.

In Figure 3.11 an example for the cyclic propagation of one refresh event in a three-place (*n=3*) topology is given.

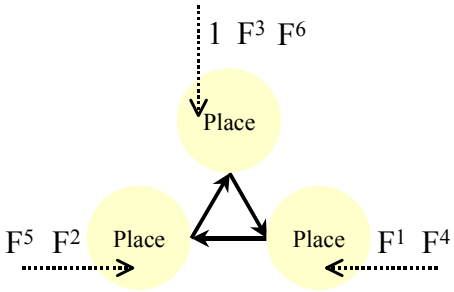


Figure 3.11. Infinite Cycles of one Refresh Event

When considering the equilibrium, large values are assumed for *t*. Thus $r(t,p_x)+q(t,p_x)$, the aggregated input to the place of the agent, is approximated by $(A/T)/(1-F^n)$. All other places receive the input weakened by the propagation from $p_x$. Hence the following equation gives the equilibrium for each place in the loop:

$$B_i = F^{|x-i|} \frac{A}{T} \frac{1}{1-F^n} \frac{1}{1-E} \tag{3.37}$$

### 3.2.4 Multi-Pheromone Analysis

The formal model does not provide any extra support to analyze agent scenarios with multiple pheromones. For each pheromone one model must be instantiated. In a multi-pheromone scenario all models share the same set of places, and each two models either have the same propagation relation, or the relations are inverse to represent upstream and downstream links.

To discuss the strength of different pheromones at the same place, input patterns to all pheromones and all places are defined first. In the discussion, the whole set of pheromones or just a subset is considered. The collection of all pheromone strength values at one place in a multi-pheromones scenario is called the mix at the place. Subsets of the pheromones are considered in sub-mixes.

Mixes or sub-mixes of pheromones are analyzed when the ratio of their strength values transmits application specific information. Then, there is information in the absolute and in the relative pheromone strength. In most applications it is only a sub-mix of pheromones jointly transmitting some data.

As an example for the transmission of data in a mix of pheromones the communication of the current mix of agent-states is considered. Assume that there are *n* different pheromones $\{ g_1, \ldots, g_n \}$ managed in the *PI*. The mix of these pheromones is evaluated at a singular place $p_s$ in the environment (no propagated input).

Let there be a set of *n* arbitrary agent-states $Z = \{ z_i : 1=1..n \}$. Each agent located at $p_s$ is in one of these states. Assume that there are $a_i \in \aleph$ agents in state $z_i$. An agent in state $z_i$

refreshes the pheromone $g_i$ regularly. The refresh behavior is governed by the global parameters $A$ (refresh strength) and $T$ (refresh rate), which apply to all agents.

For each pheromone, a separate formal model is constructed. On the basis of the model, the fixed point of the pheromone strength at $p_s$ is predicted. Let $E_i$ represent the evaporation parameter of pheromone $g_i$ and furthermore, let $B_i$ denote the equilibrium of pheromone $i$ at $p_s$ given by:

$$B_i = a_i \frac{A}{T} \frac{1}{1 - E_i} \tag{3.38}$$

With the additional requirement $\forall i, j (i \in [1,n] \wedge j \in [1,n] \rightarrow E_i = E_j)$ fulfilled, the relative number of agents in state $z_i$ at any time $t \in \aleph$ is represented in $B_i \Big/ \sum_{i=1}^{n} B_i$ if the pheromones have reached equilibrium.

### 3.2.5 Load Balance – A Real-World Example

The problem at hand is the control of an inflow of agents into a place. A similar problem arises for the guided manufacturing control system presented in Chapter 4.

The environment of the *PI* comprises two places, $p_1$ and $p_2$ (Figure 3.12). The places are linked and $p_2$ is the downstream neighbor of $p_1$. The example considers one pheromone type called "Resistance" that carries no additional data-slots. Hence, there is only one "Resistance"-pheromone. $P_R$ denotes the pheromone in the following. The propagation direction of $P_R$ is upstream. The formal model for $P_R$ specifies an evaporation parameter $E_R$ and a propagation parameter $F_R$.

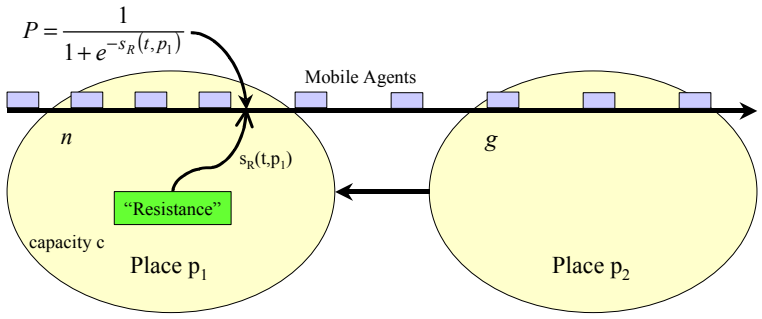


Figure 3.12. Problem Description

There is a flow of mobile agents passing downstream through the *PI*. Each mobile agent first enters the capacity restricted place $p_1$. The restriction states that there must never be more than $c$ ($c \in \Re^+$) agents located at $p_1$ in a unit time. After leaving $p_1$ each mobile agent enters $p_2$ and then moves out of the scope of the example. Without a capacity restriction, the strongest possible inflow into $p_1$ would be $n$ ($n \in \Re^+$) agents per unit time. It is the goal to reduce the load of $p_2$ to $g$ ($g \in \Re^+$) agents per unit time.

The following assumption on the decision process of the mobile agents enables a pheromone-based control of the flow. The mobile agents are known to sample the strength of $P_R$ at $p_1$ ($s_R(t,p_1)$). On the basis of the current strength, each agent decides either to pause or to move on as described in the following agent procedure:

1. Sample the strength of $P_R$ at $p_1$ ($s_R(t,p_1)$).
2. Probabilistically decide to pause or to continue, biased by $s_R(t,p_1)$. The probability to pause ($P$) is computed as:
   - $P = \left(e^{-s_R(t,p_1)} + 1\right)^{-1}$.
3. If the decision is in favor of moving on, continue to $p_2$, else go to step (1) in one unit time.

The "Resistance" pheromone provides the means to control the strength of the outflow at $p_1$, which is at the same time the inflow into $p_2$.

**Capacity Restricted Flow**

Assuming an infinite capacity at $p_1$, there may be an infinite number of agents present at the same time. Further assume a fixed probability $P$ for each mobile agent to remain at $p_1$ for one unit time.

In the beginning, $p_1$ is assumed to be empty. Then, there arrive $n$ agents in the first cycle. Applying the probability $P$, $n(1-P)$ agents leave and $nP$ agents stay. The next $n$ agents join these $nP$ agents. Then there are $n+nP$ agents at $p_1$. Out of this group $(n+nP)(1-P)$ agents leave and $(n+nP)P$ agents remain. Taking the argument to infinity ($t \to \infty$) sees $(1-P)*n\sum_{i=0}^{\infty} P^i$ agents leaving and $P*n\sum_{i=0}^{\infty} P^i$ agents remaining in one cycle. Reducing these expressions reveals that the flow of agents out of $p_1$ approximates the inflow of $n$ agents. The number of agents remaining at the place in one cycle stabilizes at $n\frac{P}{1-P}$. When the next $n$ agents arrive, $n/(1-P)$ agents have gathered at $p_1$.

The discussion concludes in the following statement: With a capacity-restriction higher than $n/(1-P)$, the flow of agents through the place is only delayed in time but not restricted in strength.

In a second step a capacity restriction of $c \le n/(1-P)$ is assumed. The assumption results in the following requirement on the inflow strength: $n \ge c(1-P)$. The requirement is also motivated by the following discussion. Assume a maximally filled place $p$. Of the $c$ agents available, $c(1-P)$ agents leave and $cP$ agents remain. To sustain the further outflow of $c(1-P)$ agents, another $c(1-P)$ agents have to arrive in the next cycle. Then, $p_1$ is filled up again.

The final conclusion is: With an unrestrained inflow $n$ greater than $c(1-P)$, the outflow depends on $c$ and $P$ alone. Thus, influencing $P$ if $c$ is known controls the flow.

**Predicted Flow Control**

One stationary agent located at $p_2$ realizes the flow control. The agent knows the flow-restriction goal $g$ and the capacity of $p_1$. With the ability to influence the strength of the "Resistance" pheromone at $p_1$ the agent is able to compute its own optimal refresh behavior on $P_R$ at $p_2$ to fulfill the goal (Figure 3.13).

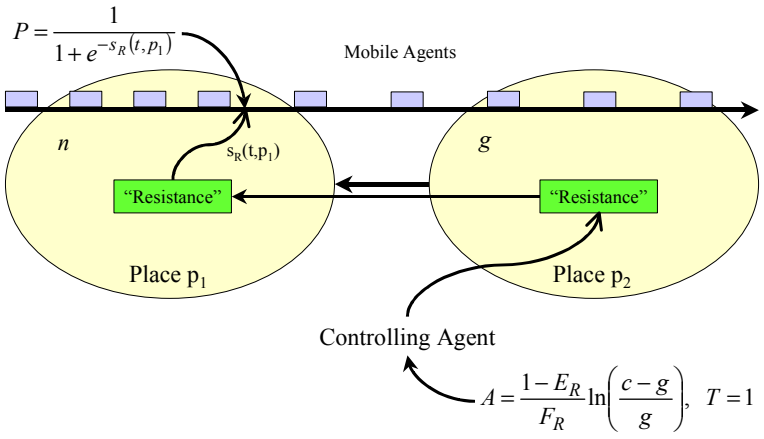


Figure 3.13. Solution to the Load-Balance Problem

The outflow *c(1-P)* out of $p_1$ has to equal *g* to restrict the inflow at $p_2$ to *g*. Therefore, the individual probability to pause must be set to *P=1-g/c*. Since *P* depends only on the strength of $P_R$ at $p_1$, the pheromone strength has to be set to $s_R(t, p_1) = \ln((c-g)/g)$.

It is not possible to set the strength of a pheromone to a specific value. But, it is possible to regularly refresh it so that it stabilizes near the required value. From the previous analysis of the *PI* it is known that a regular refresh by a strength of *A* every unit time results in a fixed point of $A/(1-E_R)$ in the sequence of the strength values of $P_R$ at the place of an agent.

The one-step propagation from the place of the stationary agents ($p_2$) to the place where the mobile agents read the pheromone is taken into account. The regular input strength required to get $s_R(t,p_1)$ to stabilize at the optimal value is computed by:

$$A = \frac{1-E_R}{F_R} \ln\left(\frac{c-g}{g}\right) \quad (\text{for } T = 1) \tag{3.39}$$

Thus, the stationary agent is able to reach its goal of a required inflow restriction without being able to actually perceive the inflow at $p_2$. It does not even have to know the unrestricted inflow into $p_1$. It only needs to know the static capacity value of $p_1$ and the probabilistic description of the behavior of the mobile agents.

## 3.3 Implementing the Pheromone Infrastructure

The following section presents a distributed implementation of the *PI*. The implementation is application-independent since the *PI* is part of the runtime environment of the multi-agent system of an application. Many of today's runtime environments already provide a communication infrastructure to their agents. But, there is no specific support for pheromone-based coordination mechanisms. To realize a pheromone-based multi-agent application in one of today's runtime environments, it is necessary to implement the *PI* as an agent system too.

Two practical extensions to the *PI* that are not covered by the formal model are introduced first. Then, the agents realizing the infrastructure are presented. Finally, requirements to the agent runtime environment are discussed.

For the remainder of the thesis, the propagation of an input to a pheromone is simply called propagation of the pheromone. In the implementation, where a pheromone is an instance of a software object, propagation is actually the transfer of such a pheromone instance.

### 3.3.1 Two Practical Extensions

The formal model presented in Section 3.2 is designed to facilitate the analysis of general features of the *PI*. Practical considerations lead to extensions of the infrastructure only introduced in the implementation, but not in the model. The extensions increase the ability of the *PI* to actually process information. Agents in the application could also realize automatic manipulation and the generation of specific views on the information. But, facilitating these operations right in the infrastructure reduces the costs of information processing and increases the applicability of the approach to a wide range of problems.

#### Direction Specific Aggregation

The first extension concerns the propagation of pheromones. Often it is necessary to know at a place from what neighbors a pheromone had been propagated. At a place with multiple upstream or downstream neighbors, from each neighbor an input could have been propagated to the place.

In the implementation of the *PI*, all pheromones carry a data slot "direction". The value in the slot is changed during propagation from one place to the next. At each place $p_x$, the value in the "direction" slot of a propagated pheromone is set to a reference to $p_x$. Thus, upon arrival at the neighbors, only the pheromones propagated from the same place aggregate. The different values in the "direction" slot may be interpreted as additional local specifications of a pheromone type.

Agents may selectively sample the strength of a pheromone for the whole place, or specific for each neighbor of the place. Thus the set of all (upstream and downstream) neighbors of a place may be considered in terms of places where the pheromone had been propagated from, and in terms of those from which no propagation occurred.

#### Application Specific Filtering

The second extension of the *PI* in the implementation introduces an automatic application-dependent change of pheromones during propagation. The filtering mechanism is very powerful as it departs from the notion that different pheromones do not interact in the *PI*.

The simplest case of application-specific filtering is given when the propagation of a pheromone is blocked depending on the presence of specific agents at a place. In this case, arriving propagation still changes the local strength values. But the pheromone is not propagated any further regardless of the availability of neighboring places.

The propagation of a pheromone may be inhibited by other pheromones. There could be threshold values to the strength of inhibiting pheromones specified: If the strength of these pheromones is stronger than a threshold value, the propagation is blocked. A continuous version of the filtering mechanism specifies an inhibition function that reduces the

propagated strength (in addition to *F*) depending on the local strength of the inhibiting pheromones. Inverse to the inhibition of propagation, the propagated pheromone strength could be reinforced through the local presence of other pheromones.

Finally, the arrival of a propagated pheromone may trigger the generation of other pheromones. Thus, an actual input to a pheromone *x* could trigger a virtual input to a pheromone *y* further down.

The application-specific filtering may be realized by local propagation rules that are executed at arrival or departure of the propagated pheromones. A direction specific aggregation as introduced in the previous section may be realized by such a departure-rule too.

With the introduction of local changes to the propagated pheromone strength, the results gained in the analysis of the formal model do not apply to the implementation as such. Especially the reinforcement of the propagation puts the general stability of the infrastructure at risk. Application programmers have to be careful when applying propagation filters.

The implementation of the *PI* allows the application programmer to define two sets of filters specific for each pheromone type. The first set is applied to incoming propagation events. The second set applies to outgoing propagation. The order, in which the filters of each set are applied, may be specified as well.

### 3.3.2 Agent-Based Implementation

The agent-oriented implementation of the *PI* is message-based and event-driven. Each place of the infrastructure is represented by a Place-agent. The computations required to operate the infrastructure may be distributed over a network of processing nodes in the same way the agent system of an application is distributed. The agents of an application interact with this part of their environment as they may interact among themselves.

In the following, a Place-agent and its place are used as synonyms. Furthermore, the general use of “agent” stands for the agents of the actual application in distinction to the Place-agents, which are “just” part of the runtime environment of the application.

#### Knowledge and Responsibilities

Each Place-agent has the references to all its upstream and downstream neighbors. The knowledge is stored in the topology management of the agent.

To get access to a place in the *PI*, the agents of an application register with the respective Place-agent. All agents registered with a place are stored in the local agent management. Only agents registered with the place may request its services.

A Place-agent manages pheromones. For each pheromone type of an application, the pheromones are stored and evaporation and propagation is realized. The registered agents are permitted access to the pheromones. A software object represents a pheromone type in the infrastructure. The object has value slots for the data of the pheromone type, and for the strength and the direction. The methods provided by the object realize the evaporation mechanism and the access to the data of a pheromone.

A Place-agent implements the pheromone evaporation asynchronously in an as-needed fashion. The continuous change of the pheromone strength is triggered by events. Reducing the computational load on the infrastructure, the current strength value is computed only when a new input to the pheromone arrives or when an agent samples the pheromone.

The access to a pheromone is pattern-based. For each pheromone type patterns may be defined that constrain the matching values in a data slot. Also, an open match to any value ("don't care") may be specified in a pattern. The match is realized in methods of the software object of the respective pheromone type.

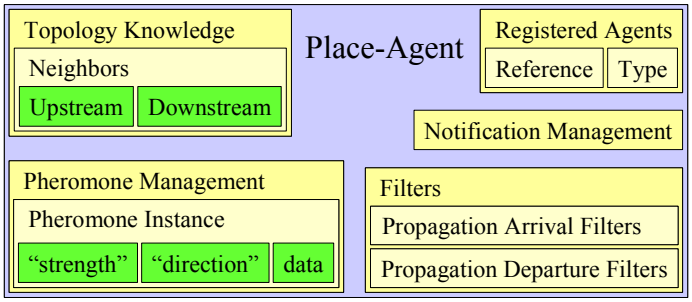


Figure 3.14. The Structure of the Knowledge of a Place-agent

Not every possible pheromone for each pheromone type is present in the pheromone management of the Place-agent. Reducing the precision of the implementation in comparison to the formal model, each pheromone type specifies a positive threshold value "precision level". A Place-agent erases a pheromone from its pheromone management when the absolute value of the strength has fallen below the threshold. The same threshold halts further propagation of a pheromone.

A Place-agent applies all filters in the propagation of pheromones. The filters come in filter-objects, programmed by the designer of the application.

Figure 3.14 illustrates the structure of the knowledge of a Place-agent.

**Service: Access and Topology**

Before an agent may request any of the services of a place, it first has to register with the Place-agent in a "RegisterAgentForPlace" message. The message takes the reference to the agent and its type. The Place-agent stores the data and checks if any notifications (see „Service: Notification" in this section) result from the registration.

An agent should be registered with only one place at a time. To gain access to services of another place, the agent deregisters from its current one. A message "DeregisterAgentFromPlace", containing the reference to the agent, is sufficient. Upon receiving the message, a Place-agent again checks for resulting notifications.

If an agent is registered with a Place-agent, it may request the local topology information. The message sent in reply to "RequestTopology" contains the references to all the neighbors of the place and an indication if the neighbor is upstream or downstream. Thus, the agents may maneuver through the *PI* dynamically.

**Service: White Pages**

The *PI* sets up a spatial structure to the multi-agent system. The system is dynamically split into groups of agents according to their current mapping to places. To support a spatial structuring of potential direct agent-to-agent communication, agents should only communicate directly inside their place.

A simple way of realizing the restriction of direct communication is to refuse the agents any system-wide White Pages service. In dynamic multi-agent systems any communication is thereby disabled since an agent needs a reference to its potential partner to send a message. Without a global White Pages service, a local service provided by the respective Place-agent still enables local communication. It is up to the designer of the application to ensure that departing agents do not take the locally received references with them.

The White Pages service is accessed with a "RequestAgents" message. In reply a place sends the set of references to the currently registered agents and their application-types.

**Service: Pheromones**

An agent registered with a place may generate an input to a pheromone by sending a "PutPheromone" message to the place. The message contains an instance of the software object that represents the pheromone type. All data slots are set to specify the pheromone. The "strength" slot is set to the strength of the input. The "direction" slot is initialized with a reference to the agent that generated the input.

Upon arrival of a "PutPheromone" message, a Place-agent extracts the pheromone from the message. If the pheromone is already present in the pheromone management, the strength of the newly arrived pheromone is added to the (correctly evaporated) strength of the managed pheromone. The aggregation only applies to pheromones with all but the "strength" slot set to the same value. A difference in the "direction" slot already prevents an aggregation.

If there is no managed pheromone matching the new pheromone, it is put directly into the pheromone management.

If the propagation parameter of the respective pheromone type is not zero, propagation to the neighbors of the place commences. On the basis of the propagation direction of the pheromone type, the Place-agent first determines the set of neighboring places, to which the pheromone is to be propagated. In a second step, for each neighbor a new pheromone is created. The strength values are set according to the weakening by the propagation parameter and the division by the number of neighbors (Equation 3.1). After all relevant filters have been applied, the pheromones are sent to the neighboring places, again using the "PutPheromone" message.

"PutPheromone" messages arriving from agents and not from other places may trigger the notification of local agents.

The registered agents may also access the locally managed pheromones. A "GetPheromones" message contains patterns. The Place-agent determines all currently available pheromones that match the patterns. For each matching pheromone a copy is created, which carries the current strength and the additional data. The resulting set is sent to the agent, from which the request came.

Compared to the other data slots, the “direction” slot in a pattern is handled differently in the matching process. If a specific direction (reference to a place) is set in a pattern, only completely matching pheromones are selected. But, if any direction is accepted, all pheromones matching the other slots are selected first. Then, the strength of the pheromones is added up. The “direction” slot of the resulting aggregated pheromone is set to a reference of the local place.

**Service: Notification**

Most pheromone-based agent algorithms specify agent activities in response to specific events in the *PI*. Such events are the arrival or departure of agents, or an input to a pheromone.

Agents registered with a place may request a notification in case of the occurrence of specific events. The messages “RegisterForAgentEvent” and “RegisterForPheromoneEvent” enter an agent in the notification management of the place. The agent type and whether an agent arrival or departure is expected specifies a notification request in case of agent movement. The type of the pheromone specifies a notification request in case of an input.

The notification is given using the “AgentEventOccured” and “PheromoneEventOccured” messages. The messages specify the event occurred, giving reference to the actual agent, or containing the generated pheromone respectively.

A place handles the respective deregistration messages too.

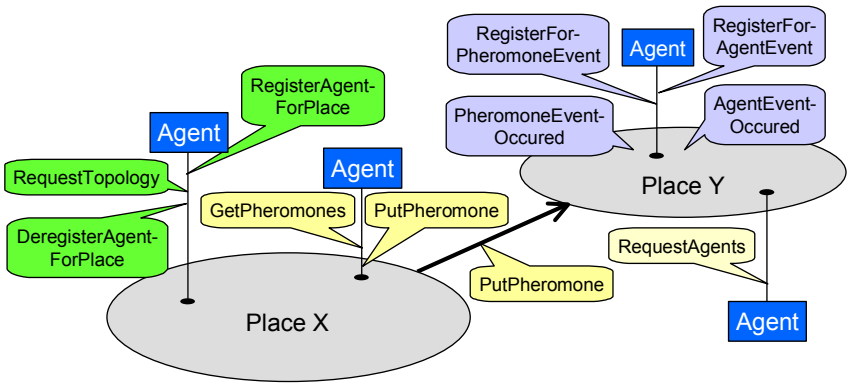


Figure 3.15. The Services of the Place-agent

The messages sent to a Place-agent to request its services are shown in Figure 3.15.

### 3.3.3 The Agent Runtime Environment

The agent-based implementation of the *PI* assumes features of the agent runtime environment that concern the internal architecture of the agents, the communication infrastructure, and the execution capabilities. Furthermore, there are features required for an effective execution of real-world sized applications.

It is assumed that the internal agent architecture supports reactive agent behavior. A Place-agent is never proactive. It only reacts to the incoming requests. Since a Place-agent does not have any physical abilities in the real world (sensors, actors), its reactiveness is

directed towards inter-agent communication only.

The second assumption taken is the ability of an agent to handle internal data. It is not just required that the agent may operate on large sets, it has to cope with sets of variable size too. It is assumed that the internal data management is object oriented, operating on object hierarchies defined by the programmer of the application.

The chosen message-based implementation of the interface between the agents and the *PI* requires a fast and efficient agent-to-agent communication. Sending a message has to be "cheap" in terms of transmission time and resource usage.

It has to be assumed, in a general-purpose agent runtime environment, that there is a homogeneous address-space where each agent is uniquely referenced. In principle it has to be possible to communicate with any agent, regardless of the location of its processing node. The distribution of the agent execution to several processing nodes should be transparent on the reference-level.

Messages sent among agents have to carry software objects and sets of objects. The designer of an application defines these objects. During communication not a reference to an object, but the object itself is transmitted. Thus, the receiver really owns the received object by having the sole reference to it.

In terms of execution capabilities the ability of the runtime environment to handle many lightweight agents is assumed. The assumption states a general requirement in pheromone-enhanced synthetic ecosystems. It does not just hold for the specified implementation of the *PI*. Real-world sized applications may comprise hundreds or even thousands of agents.

Following a pheromone-based approach in real-world sized agent systems also requires a specific structure of the distribution of the agents to processing nodes. The same processing node should execute agents located at the same place in the *PI*. With the restriction of direct agent-to-agent communication to interactions where both agents are registered with the same place, the global bandwidth needed in the execution of the system is drastically reduced, assuming that communication on one processing node does not take up any actual bandwidth on the network (re-using bandwidth).

With the aim to have agents of the same place executed at the same processing node, the execution of agents has to be mobile if the agents themselves are mobile too. An agent may deregister from its current place at any time and register with another place. If the agents at the new place are computed at a different processing node, then the virtual movement in the *PI* should be mirrored in an actual change in the distribution of the agent system. The requirement holds especially for fielded applications, operating in real-time.

The *PI* implemented by Place-agents has been realized prototypically in the JAVA-based system DARE ("Distributed Agent Runtime Environment") of the Multi-Agent Systems department at DaimlerChrysler AG - Research and Technology 3. DARE provides all the presented requirements except the dynamic relocation of the execution of agents. Using a general implementation of the *PI*, it was possible to evaluate the results of the formal analysis in different scenarios. The guided manufacturing control system, presented in Chapter 4, is implemented on the basis of DARE and the Place-agents too.

# Chapter 4 - Guided Manufacturing Control

*Chapter 4 presents a stigmergetic synthetic ecosystem for manufacturing control. In the design of the system the design principles (Section 3.1) have been followed. The pheromone infrastructure is deployed to support the implementation, the tuning, and the evaluation of the software system (Sections 3.2 and 3.3).*

*First, Section 4.1 introduces the chosen manufacturing control domain and derives the spatial structure of the agent system. Then, the components of the software architecture are listed. These components, the sub-systems and their internal layers, are presented in the remainder of the chapter.*

*The reactive layer of the control sub-system is presented in Section 4.2, the tasks and agents of the singular interface layer are specified in Section 4.3, and Section 4.4 introduces the strategy implementation layer of the advisory sub-system. The resulting system-level behavior is discussed in a small but realistic demonstration in Section 4.5 where all the previously presented components are evaluated.*

## 4.1 Domain, Structure, and Architecture

On the basis of the rationales for holonic manufacturing as they are presented in the background Section 2.2.2, the agent system presented in the following combines distributed and reactive control with optimization according to global production goals. The combination is achieved bottom up, leaving the decisions that directly change the physical environment to the localized control, but guiding it with advice given under explicit global consideration. Hence, the agents make up a guided manufacturing control (*GMC*) system.

The *GMC* system is designed for flexible flow shops (Section 2.2.1) in industrial mass-production. The most important characteristic of a flexible flow shop is that of all possible material flow patterns only a very small sub-set results in a globally optimal performance of the production process. The current optimal sub-set of flow patterns strongly depends on the situation in the manufacturing system and it usually cannot be determined beforehand.

There is a specific flexible flow shop, onto which the *GMC* system is primarily designed: the final steps of painting passenger cars at the Mercedes-Benz car plant of the DaimlerChrysler AG in Sindelfingen (Germany) provide the application background. Manufacturing specific design decisions have been taken to fit this flow shop and the demonstration and evaluation in Section 4.5 is based on a segment that is characteristic for the paint shop. Therefore, to give an understanding of these characteristics, the paint shop is briefly introduced first.

### 4.1.1 The Paint Shop

Figure 4.1 plots a process graph that applies to car bodies, passing the last two processing steps in the paint shop in the plant. Structurally, the process graph is similar to

those of other paint shops in the automotive industry.

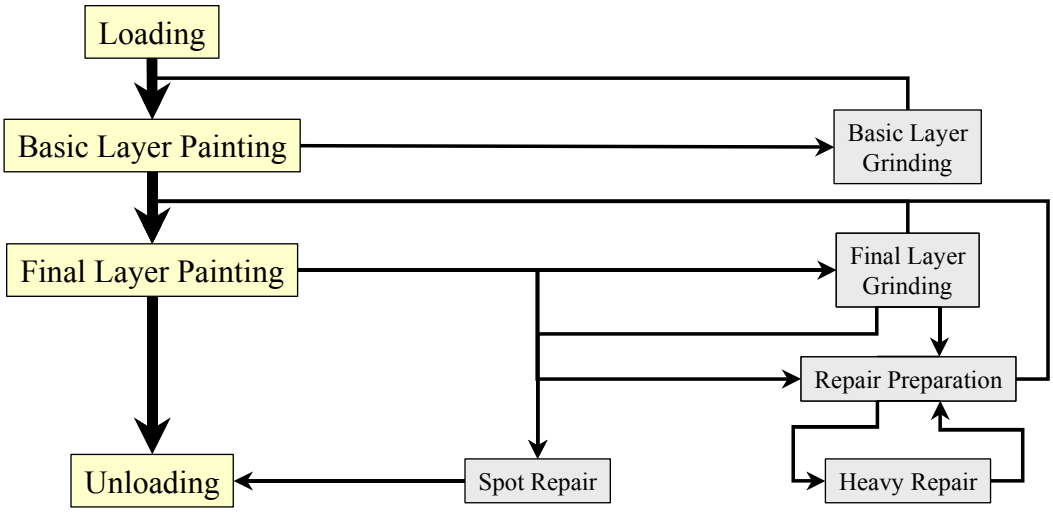


Figure 4.1. Process Graph of the Paint Shop Segment

The last two steps in the painting of car bodies require the highest flexibility. First, the cars have to successfully pass “Basic Layer Painting” (BLP) and then they have to get through “Final Layer Painting” (FLP). Whereas BLP paints only three different types of basic color (light, medium, dark), the FLP process provides a choice of more than twenty colors in an automated painting process.

A car entering this section of the manufacturing system represents a specific order already and hence the required final color is fixed. There is a relatively small set of frequently occurring colors, but most colors are seldomly requested. The volume and the mix of orders always change with the changing demand on the market.

The complexity of the considered manufacturing control problem stems from the cyclic character of the process graph of a car. Not every painting process is successful. Most cars emerging from the exit of a BLP unit are permitted to go on to one of the FLP units. But some have to get a “Basic Layer Grinding” job performed and then they must reenter a BLP unit. The relative strength of these different output streams over a fixed period defines the yield of a BLP unit.

A FLP unit has a yield too. Successfully painted cars may leave the system. But the other cars require repair tasks like “Spot Repair”, “Repair Preparation”, “Final Layer Grinding”, or even “Heavy Repair”. Except cars emerging from a “Spot Repair” unit, all these cars reenter the FLP units after the repair task.

To provide capacity, flexibility, and redundancy, there is always more than one processing unit able to perform the required processing step. It is the task of the control system to make efficient use of the available flexibility. A highly connected layout of the transport system linking the processing units supports the effort.

The transport system in the BLP and FLP segment of the paint shop comprises standardized stationary transport elements. Currently, there are more than 500 elements of different types, mainly conveyors and lifts. With the exception of the sorting buffers, which provide random access to create batches of cars of the same color, all buffers are just specific zones in the transport system made up of standard transport elements.

### 4.1.2 The Spatial Structure

First, the layout of the virtual space where the agents of the *GMC* system operate in is defined. The layout is represented in the *PI* and it is derived from the spatial structure of the underlying manufacturing system, which comprises transport elements, processing elements, and workpieces. In the paint shop transport elements are lifts, conveyors, and rotation tables, whereas processing elements are paint-booths, dryers, and other processing stations. With the focus on the material flow, the model does not represent single robots, tools, or human workforce. Every car is considered one workpiece.

The space of the manufacturing system is divided into elementary zones. A zone either covers transport elements making up buffers or crossings in the material flow, or it mostly contains processing elements and thus aggregates elementary processing steps into one abstract step. In addition, there are small-sized zones, containing loading or unloading machinery. These are the physical interfaces to the world outside of the manufacturing system.

Each elementary zone is connected to other zones in the material flow. These connections are either entries (material coming in) or exits (material sent out). The following restriction to the allocation of elementary zones holds: A zone containing processing elements is only reached through one entry and it has only one exit. The restriction supports the identification of zones where transport decisions are taken, and of zones where processing tasks are fulfilled.

Additionally, elementary zones without entries are only permitted to have one exit, and those without exits must have one entry only. Hence, there are no routing decisions taken in the loading and unloading zones.

The layout of the *PI* is specified in a one-to-one mapping of the elementary zones to places. The links among these places are set according to the connections of the zones. The downstream direction of a link is the direction of the material flow from the exit of one zone into the entry of the next one. In Section 4.5 the identification of elementary zones in a manufacturing system is discussed on the basis of the paint shop.

### 4.1.3 The Architecture

Illustrating the following presentation of the system architecture, in Figure 4.2 the structure of the system is shown, marking where agent types are introduced in the bottom-up specification and also, where their behavior is extended. The illustration starts with the bottommost part of the system, the *PI* and its Place-agents. It only includes the components presented in Chapter 4.

The *PI* constitutes a part of the environment of the agents of the *GMC* application. It provides the means for indirect communication and automated aggregation of data in an active environment to realize stigmergetic multi-agent coordination. As it is discussed in Section 3.3.2, the *PI* is implemented as a multi-agent system sharing the runtime environment with the agents of the *GMC* application.

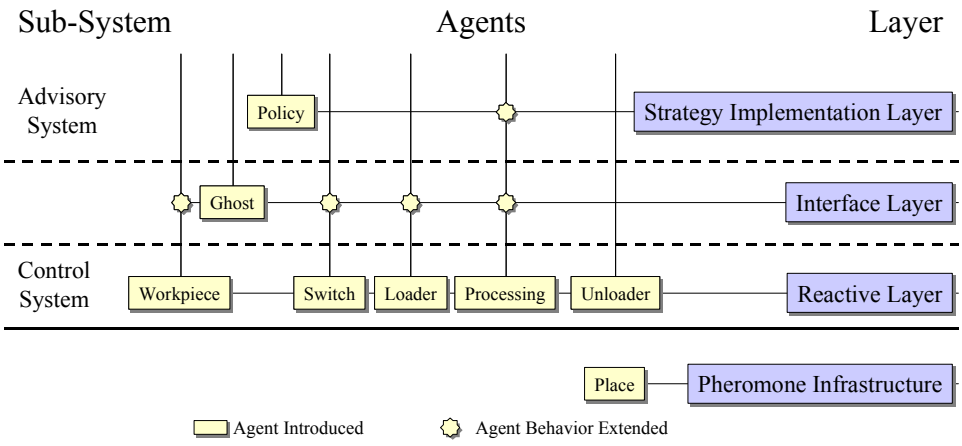


Figure 4.2. Components and their Agents

The *GMC* systems architecture is specified from bottom up. Structuring the major functionalities, the architecture comprises two sub-systems, one of which provides reactiveness and flexibility while the other one adds optimization based on the production goals. Accordingly, the first sub-system is called the control system and the second one is the advisory system.

Both sub-systems may in turn comprise several layers. Restricting the following specification to basic functionality only, the reactive layer of the control system and the strategy implementation layer of the advisory system are presented and evaluated in detail. Concepts for additional layers of the advisory system and for a third sub-system are discussed in Chapter 5.

An interface layer links the two sub-systems. It does not provide any additional functionality to the (human) operator of the *GMC* system, but it facilitates the operation of the advisory system by aggregating information and it incorporates advice into the control system.

The components of the *GMC* system contain a large number of agent types. In the bottom-up specification for each agent type there is one layer where its basic functionality is defined. At higher levels of the architecture (layers or sub-systems) the basic functionality of an agent may be extended to provide additional services.

There is a minimal configuration of the *GMC* system. The reactive layer of the control system alone is already sufficient to operate the manufacturing system. It guarantees that every workpiece sent into the system is eventually processed according to its requirements. The other layers and sub-systems only provide optimization of the basic operation without compromising the fulfillment of hard constraints.

### 4.1.4 Performance Evaluation

Before the structure and the mechanisms of the *GMC* system are presented in more detail a short remark on the evaluation of the performance of a new manufacturing control system should be made. Later sections propose an approach to model and analyze the agent system that eventually may provide qualitative or even quantitative predictions of the emerging system behavior. But in general today's manufacturing processes are much to complex to be formally modeled in the required granularity.

The only alternative to formal modeling short of physically installing the control system on the factory floor is to create a realistic simulation model of the desired production processes to which a manufacturing control system may interface. In a first step of the evaluation process, an implementation of the current control approach should provide the baseline against which every new approach has to measure up. This step is only feasible if the production process can still be handled by conventional manufacturing control systems. Completely new processes might mandate new control approaches.

In the following steps, instantiations of the new control system are linked to the simulation. In all simulation runs the same performance indicators are observed and protocols of their value over time are created. Performance indicators of the production process are for instance local and global throughput values, the load of processing stations and buffers, or the work in progress (WIP) and its composition in the system. The evaluation should not be restricted to production performance parameters alone. Performance parameters of the control system, like computation and communication load or required human interference, and the required effort for the design, implementation, tuning, and maintenance of the system should be included into the evaluation.

## 4.2 Control System: The Reactive Layer

The reactive layer of the control system is the core of the *GMC* system. Therefore, it is responsible for the fulfillment of all hard constraints for the production operation. The agents of the layer enable the manufacturing of products. There is no optimization implemented into their basic functionality.

The agents making up the reactive layer are identified on the basis of the layout of the *PI* following the PROSA reference architecture (Section 2.2.3). They are introduced in terms of their responsibilities, their knowledge, and their behavior. The specification of the reactive layer concludes with a high-level integration of the behavior of the agents.

### 4.2.1 Entities and their Behavior

The PROSA reference architecture suggests three basic agent types for manufacturing control: the Resource-agent type, the Order-agent type, and the Product-agent type. Resource-agents represent the resources of the manufacturing system, controlling the physical handling of the workpieces. Order-agents are identified with requests to the manufacturing system. Product-agents provide process knowledge for the different product variants produced.

The specific Resource-agent types are derived from the structure of the *PI* and the characteristics of the underlying elementary zones. There is exactly one Resource-agent for each place.

Loader-agents map to places whose zones contain the loading machinery, and there are Unloader-agents at places of the unloading machinery. These places either have only one exit (Loader-agent) or one entry (Unloader-agent). Transport decisions are taken at places with multiple entries or multiple exits. These places are occupied by Switch-agents, routing the material flow from the entries to the exits. Finally, Processing-agents are located at places where processing elements dominate the elementary zone. A Processing-agent represents an aggregated processing step taken while a workpiece passes its zone.

Each Resource-agent has complete control of all its local elements of the manufacturing system. Local elements are located in the elementary zone that is mapped to the place of the agent.

In the paint shop an order to the manufacturing system maps to exactly one workpiece (car). Hence a Workpiece-agent is an Order-agent. Other applications with complex orders (e.g. in assembly) might require the introduction of an aggregation of Workpiece-agents into higher-level Order-agents.

Each Workpiece-agent has a place in the *PI* determined by the physical location of the workpiece of the agent. The elementary zones represent a clear-cut subdivision of the manufacturing system and thus the position of a workpiece always coincides with exactly one zone. The place mapped to the zone is where the Workpiece-agent resides. As a consequence, each Workpiece-agent always occupies the same place as the Resource-agent handling its workpiece. Hence, direct interaction may commence without compromising locality.

The reactive layer does not include any specific Product-agents, restricting the *GMC* system to applications with only a relatively small number of product variants and slowly changing processes. The paint shop fits these restrictions. Process knowledge is directly incorporated into the Processing-agent and into the Workpiece-agent. They share the responsibilities of a Product-agent.

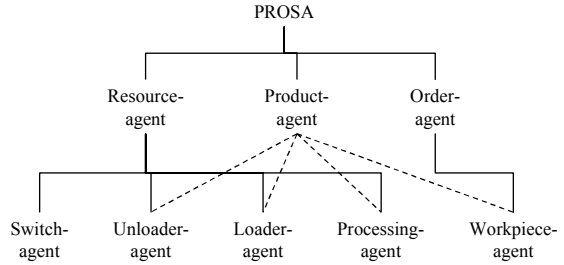


Figure 4.3. Reactive Layer Agents as Derived from PROSA

Figure 4.3 illustrates the agent types of the reactive layer with their responsibilities derived from PROSA. In the following, each agent type is presented, specifying its responsibility, its knowledge, and its behavior.

### The Loader-Agent Type

A Loader-agent represents a physical entry into the manufacturing system. Through the entry workpieces are loaded into the transport system. It is assumed that the *GMC* system is able to decide when a workpiece is loaded and that the Loader-agent may select the next workpiece according to a specific product or variant.

The inflow at the entry is specified by two parameters: the product mix and the volume of inflow. A Loader-agent may influence both in its loading operation by varying the products and the moment they enter the system. The behavior of a Loader-agent is specified by the following cycle:

1. Select next product.
2. Attempt to load the respective workpiece.

3. If the attempt was successful, create a Workpiece-agent and hand over the responsibility for the loaded workpiece.
4. Wait in accordance to the current loading rate and then restart at step one.

The loading of a workpiece may fail for two reasons: Either the exit of the loading machinery is blocked, or there are no workpieces of the required product available.

There is no interaction between the Loader-agent and other agents besides handing over of the responsibility for the workpiece. All decisions are taken by the Loader-agent alone, based on static information available to the agent.

### The Switch-Agent Type

A Switch-agent is responsible for one elementary zone in the transport system. Its zone has a set of entries and exits and a routing structure in-between. Because of possible internal routing restrictions inside the zone, a workpiece entering the zone through a specific entry has access to only a subset of all exits.

The Workpiece-agent tells the Switch-agent, where to take its workpiece. If such information is not given, the Switch-agent awards a default exit that is specific to each entry.

The activity of a Switch-agent is triggered by two external events. First, there are incoming requests by the Workpiece-agents, registering their workpieces for a transport to a specified exit. The Switch-agent stores the information together with a reference to the agent for a fixed period.

The second trigger-event is the arrival of a workpiece at an entry of the Switch-agent. In this case, the Switch-agent determines the required exit for it. Either the exit is specified in a request previously received by the agent of the workpiece, or the default exit for the respective entry is selected. Eventually, the Switch-agent realizes the physical transport of the workpiece through its zone.

Whenever alternatives in the sequencing of workpieces arise during transport, the selection is taken randomly. Priorities or other additional workpiece parameters are not considered. Alternative internal routes are also selected regardless of the workpieces.

With the arrival of a workpiece at an exit, the task of a Switch-agent is completed. If the Switch-agent knows the agent responsible for the workpiece, it tells the agent the place in the *PI* the exit is linked to, so that the agent may follow its workpiece in virtual space.

### The Processing-Agent Type

A Processing-agent is responsible for an elementary zone in the manufacturing system where the processing state of passing workpieces is changed. It aggregates the processing activities in the zone into one abstract processing step, representing the processing ability of the local resources that may be parameterized. The Processing-agent manipulates a workpiece as requested by its agent.

Activities of a Processing-agent are triggered by incoming requests of Workpiece-agents and by the arrival of their workpieces. In a request the Workpiece-agent asks the Processing-agent to apply its processing capability with a given set of parameters to the

workpiece. The request is stored together with the reference to the Workpiece-agent. When the workpiece arrives, the Processing-agent knows what processing is to be performed with it.

If a workpiece arrives at the zone of a Processing-agent without having been registered beforehand it is either passed through the zone without being processed. If passing is not possible, the entry remains blocked and the Processing-agent has to handle the internal disturbance of the control system. It could, for instance, try to find out the responsible Workpiece-agent, or it could inform the administrator. Behavior safeguarding stability is required for the fielded implementation and it takes an important place in the design effort. But, it is not essential for the demonstration of the basic functionality and therefore it is not considered here any further.

It is assumed that the Processing-agent is able to sense the new processing state of the workpiece at the end of the processing. When the workpiece has reached the exit, its agent is told the new state and the next place in the *PI*. The task of the Processing-agent is fulfilled.

Before processing workpieces, the Processing-agent has to attract them first. The attraction is realized by pheromones. The reactive layer defines a pheromone type "Ability" ($P_A$). Besides "strength" and "direction", $P_A$ specifies one additional data-slot "ability" where the processing ability of a Processing-agent is stored. Extensions to the *GMC* system may introduce additional data-slots, specifying, for instance, parameters to the respective ability. Inputs to $P_A$ propagate upstream against the flow of the workpieces. The propagation is stopped one place before the next upstream Processing-agent. The restriction, realized by a propagation filter in the *PI* (Section 3.3.1), is required to handle cycles in the layout.

Each Processing-agent regularly generates an input to $P_A$ pheromones at its place. It refreshes the pheromone matching its current processing ability. These inputs propagate upstream, thereby creating a flow field for Workpiece-agents to follow in case they require the respective processing.

### The Unloader-Agent Type

The behavior of an Unloader-agent is very similar to that of a Processing-agent. Responsible for one exit out of the manufacturing system, it attracts workpieces. Therefore, it regularly refreshes the $P_A$ pheromone that carries "unloading" in its "ability" slot.

When a workpiece arrives at the entry of the elementary zone and it is registered for unloading, the agent takes it out of the system. After completion of the unloading process, the responsible Workpiece-agent is informed.

### The Workpiece-Agent Type

A Workpiece-agent is responsible for one workpiece while it is transported and processed in the manufacturing system. It is created when the workpiece is loaded by the Loader-agent. Its task is completed when an Unloader-agent takes the workpiece out of the system. The decision loop of a Workpiece-agent is depicted in Figure 4.4. It is one continuous cycle with several alternative branches, advanced by the ongoing transport and

processing of the workpiece.

Workpiece-agent knows the current processing state of its workpiece. A change in the state is communicated from the Processing-agent that has caused it. The agent knows the process plan of its workpiece because of its Product-agent functionality. But a Workpiece-agent does not plan ahead. Instead, it follows the late commitment philosophy; taking decisions only when they are due while keeping options open as long as possible.

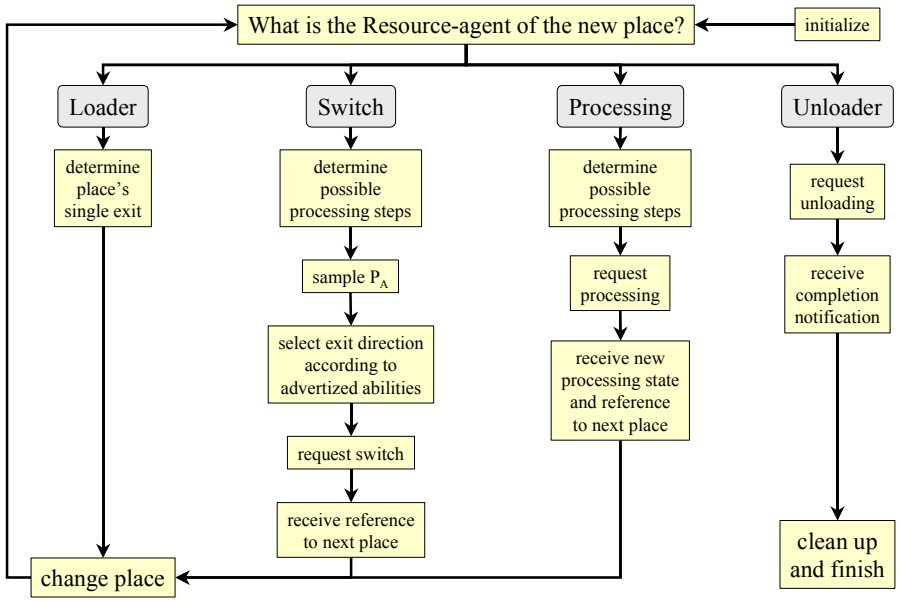


Figure 4.4. The Behavior of the Workpiece-agent in the Reactive Layer

A Workpiece-agent mirrors the movement of its workpiece through the manufacturing system in its moves through the *PI*. While it moves, it executes the following decision loop, starting with the arrival at a new place:

<u>If it has arrived at a place of a Loader-agent:</u>

1. Find the only downstream place

<u>If it has arrived at a place of a Switch-agent:</u>

1. Determine all possible next processing steps from the process plan
2. Sample $P_A$ pheromones specific to the exit-directions
3. Determine the set of downstream directions, abilities to fulfill one of the next processing steps have been propagated from
4. Randomly select one of these directions
5. Request transport of the workpiece into the selected direction from the Switch-agent
6. Receive reference to next place from Switch-agent after transport was completed

<u>If it has arrived at a place of a Processing-agent:</u>

1. Determine all possible next processing steps from the process plan
2. Request all these steps from the Processing-agent
3. Receive new processing state and reference to the only downstream place from Processing-agent after processing was completed

4. Update internal processing state

If it has arrived at a place of an Unloader-agent:

1. Request “unloading” from Unloader-agent
2. Receive information about completed unloading
3. Fulfill any cleanup or archiving task and finish

Move to the next place as determined and restart loop

Following this algorithm, the Workpiece-agent takes local routing and processing decisions based on only locally available information and its internal state. The operation does not require any communication with agents outside of the current place of the Workpiece-agent in the *PI*. It respects the autonomy and expertise of the Resource-agents.

### 4.2.2 The Emerging Behavior

The manufacturing system is divided into elementary zones, each controlled by a Resource-agent. There are Switch-agents realizing the transport of workpieces through their zones in the transport system. There are Processing-agents able to transport workpieces too, but their focus is on the processing of workpieces. Finally, there are Loader-agents and Unloader-agents physically interfacing the manufacturing system with the outside world.

The structure of the physical system is mirrored in the *PI*. For each Resource-agent one place exists where the agent resides. Located at the same place there are the Workpiece-agents representing workpieces in the zone of the Resource-agent. Direct interactions only take place among agents located at the same place.

Whereas the Resource-agents are stationary in the *PI*, the Workpiece-agents move from place to place. They follow the physical transport of their respective workpiece through the manufacturing system. A movement of a Workpiece-agent is triggered by a notification that the workpiece has been passed on into the next zone. The notification comes from the local Resource-agent.

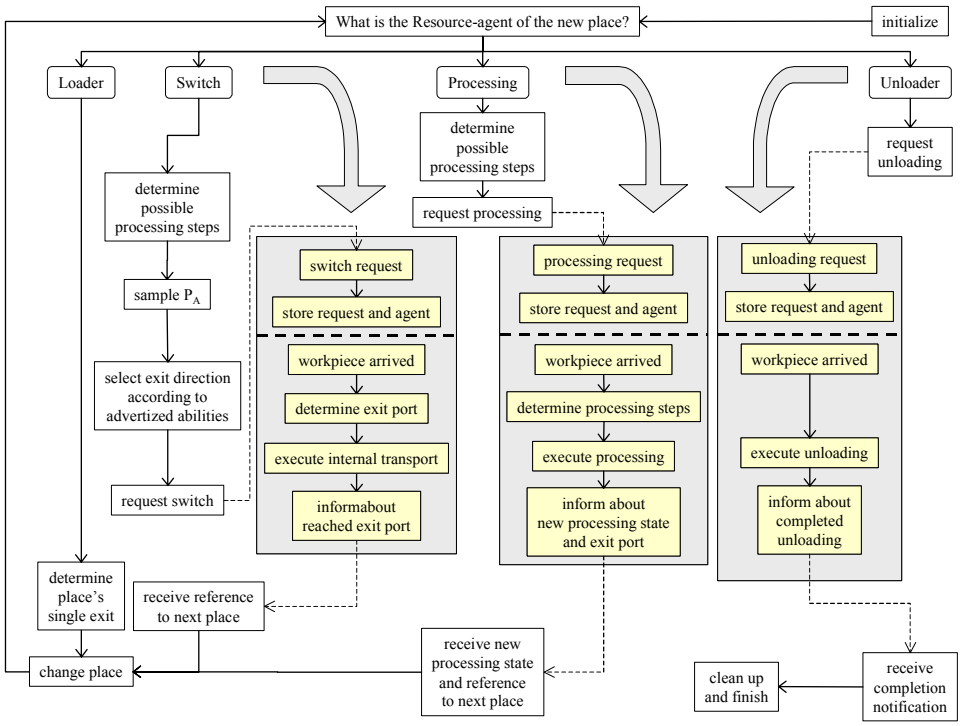


Figure 4.5. The Activities of a Workpiece-agent Trigger Resource-agents

Only zones of Switch-agents have more than one exit. Hence, routing decisions are taken only at the place of a Switch-agent. A routing decision is taken by a Workpiece-agent and then executed by a Switch-agent taking the workpiece to the selected exit. The exit is chosen on the basis of the current processing state of the workpiece, the process plan of the product, and the local flow fields of $P_A$ pheromones.

The global routing behavior emerging from these local decision processes takes the late commitment approach to the extreme. Workpiece-agents never actually commit to a processing resource. With every routing step they just reduce their options to those offered by one of the available local directions. The routing mechanism is always open to resources coming on-line or going off-line.

Figure 4.5 illustrates how the behavior of the Workpiece-agent and the transport of its workpiece trigger the activities of the Switch-agent and the Processing-agent. The processes on shaded background are the ones run by Resource-agents. A Workpiece-agent and Resource-agents synchronize in the handling of the workpiece, dynamically creating two-agent teams to fulfill the arising transport and processing tasks.

$P_A$ provides local information on the processing abilities of processing resources, separate for each downstream direction of a place. Each downstream direction corresponds to an exit of the respective zone in the manufacturing system. Selecting a downstream direction is equivalent to choosing an exit. When a workpiece has eventually passed an exit, the Workpiece-agent moves to the corresponding downstream neighbor. In Section 4.5.2 the emergence of spatial patterns in $P_A$ is discussed in the paint shop demonstration.

Evaporation of $P_A$ pheromones guarantees that a change in the availability of downstream processing resources eventually becomes known at the relevant places in a change of the local flow field. The dynamics of these changes and the constraints of the manufacturing system have to be taken into account when the evaporation parameter of $P_A$ and the refresh behavior of the Processing-agents and the Unloader-agents are tuned.

Information is lost too fast if evaporation is too strong. The Processing-agents and the Unloader-agents have to spend a lot of resources to keep the required attraction of workpieces sufficiently strong. On the other hand, if $P_A$ does not evaporate fast enough, too many workpieces might be attracted into the wrong direction. The delay in communication is extremely critical if a processing ability is permanently lost and the zone of the respective Processing-agent does not permit a workpiece to pass unchanged.

The formal model of the *PI* (Section 3.2.1) provides the means to tune the parameters according to the specific requirements of the underlying manufacturing system. There are results available providing predictions of fixed points in the local strength of a pheromone and the time required to reach them in specific layouts and under specific input patterns.

A Workpiece-agent takes probabilistic routing decisions. First, the set of permitted downstream directions is determined. These are all directions where a flow field of a $P_A$ pheromone exists that matches one of the required next processing steps. The selection of a permitted exit is random – an interfacing point for additional layers of optimization. Instead of an equal chance for all permitted directions, the choice might be weighted according to probabilities provided by additional reasoning. Statistically, the material flow splits and merges evenly as long as there is no congestion. Other flow-patterns are achievable with additional influence on the routing decisions. The advisory system attempts to set such an influence.

## 4.3 The Interface Layer

In-between the control system and the advisory system there is the interface layer. It is the task of the interface layer to collect information about the control system and to extend the basic behavior of the reactive layer agents to integrate the advice given by the advisory system into the operation of the control system.

The advisory system operates on material flow data, specifically on patterns in the flow of workpieces through the manufacturing system. A pattern combines sets of load values that are specific for a processing state at each elementary zone in the manufacturing system at a given time. The load is the number of workpieces in a given state present at a zone at that time.

The emergence of a material flow pattern depends on the inflow into the system, the currently available processing capabilities of the resources, and the yield of their processing actions. Furthermore, the routing decisions taken by the Workpiece-agents strongly influence a pattern. The interface layer has to create two patterns for the advisory system: the current material flow pattern, and a prediction of the pattern in the near future. The first part of the following section focuses on the generation of these two patterns.

Advice generated by the advisory system must be transmitted to the control system and integrated into the decision processes. The second stage of the presentation of the interface layer discusses how the agents of the reactive layer are extended so that the control system is able to heed the given advice.

### 4.3.1 Information for the Advisory System

For two reasons it is very important that the interface layer links the two sub-systems only through indirect communication through the *PI*. First, the advisory system operates on

a different time-scale than the control system. Its decision processes run slower because it reasons on aggregated flow data. If the advisory system were coupled with the agents of the control system in direct communication, the generation of advice would have to run on the time-scale of the control system too.

A second reason to decouple the two systems is the preservation of the autonomy and stability of the control system, which is the sole controller of the manufacturing system. The advisory system only provides guidance. It is up to the agents of the control system to heed or to ignore the given advice. Ignoring signs in the environment is much easier to achieve than ignoring direct communication by one interaction partner, which requires complex interaction mechanisms.

### Emergence of the Current Flow-Pattern

The current load of a zone in the manufacturing system equals the number of Workpiece-agents present at the related place in the *PI*. It is the task of the Workpiece-agents to update the load information in the environment.

As demonstrateted in the analyses of the formal model, a joint regular refresh behavior of agents located at the same place takes the strength of the refreshed pheromone to a fixed point. The value of the fixed point is proportional to the number of agents (Section 3.2.3). To generate the required pattern, each Workpiece-agent regularly refreshes a specific pheromone.

The interface layer defines a pheromone type "State" ($P_S$). In the global spatial pattern of this pheromone type, the current flow is captured. $P_S$ has one addition data-slot "state", accepting values from the set of possible processing states of the workpieces. The propagation parameter of $P_S$ is zero, preventing any propagation.

All Workpiece-agents follow the same regular refresh behavior. They always refresh the $P_S$ pheromone whose "state" slot is equivalent to the current processing state of the represented workpiece. Each Workpiece-agent generates input events of a strength specified by the global parameter "StateRefreshStrength". The rate of the input is given by the joint parameter "StateRefreshRate".

As a consequence of the refresh activity of the Workpiece-agents, at each place the current load specific for each processing state is perceivable in the $P_S$ pheromones.

### Emergent Prediction of the Future

The prediction of the near future is based on a Monte Carlo simulation approach. In Monte Carlo simulations there is a model comprising several probability distributions in a model structure. During the course of the simulation, these probabilities are sampled, following the model structure. The distribution of the simulation results is examined and the more often the simulation is run, the more correctly reflects the observed distribution the modeled system.

In the case of the pattern prediction, the system is its own model. The *PI* gives the structure of the simulation model, the agents of the *GMC* system realize the simulation, and the results are again made available in the *PI*. With the integration of the prediction model into the general operation of the *GMC* system, the model changes in step with the rest of the system. Furthermore, the prediction is always up-to-date because it is

continuously recreated based on the current situation.

The pattern of the material flow is influenced by the processing of workpieces and by the transport decisions of their agents. Predicting a future pattern requires the prediction of processing results and an emulation of routing decisions. These two processes contain the probability distributions for the Monte Carlo simulation, since neither the execution of a processing step nor a routing decision is deterministic. Therefore, these processes have to be sampled repeatedly in the simulation.

A Processing-agent handles the processing of workpieces. For the prediction of the future pattern, each Processing-agent has to predict results of processing attempts. The probable outcome of a processing step is determined by the yield of the resource. Depending on the result, a processed workpiece has to take different routes, resulting in different flow patterns. On the other hand, processing time is not relevant for the short-term prediction of the material flow.

The prediction of processing results is more up-to-date when using a short-term average yield of the processing step instead of a long term one. Therefore, a Processing-agent internally keeps a short floating average of the yield of its processing step.

In its extended behavior specified by the interface layer, a Processing-agent acts on an additional trigger. When the agent receives a request for the simulated processing of a workpiece in a specified processing state, it immediately replies with the predicted outcome. The prediction is probabilistic, determined by the floating average yield of recent processing operations.

The requests for simulation come from Ghost-agents. Ghost-agents are the only genuine agents of the interface layer. They are regularly created by Workpiece-agents. A Ghost-agent emulates a Workpiece-agent in its run through the *PI*. Right after its creation, the Ghost-agent is a perfect copy of the Workpiece-agent by which it was created. Its internal state equals the current processing state of the workpiece and it is located at the same place as its Workpiece-agent.

A Ghost-agent represents one possible future of its Workpiece-agent and it realizes one simulation run through the Monte Carlo model. Encountering a Switch-agent, a Ghost-agent samples the same pheromones and then it takes a routing decision on the basis of the same decision process followed by its Workpiece-agent. Only, the Ghost-agent does not interact with the Switch-agent, because it does not have a real workpiece to route. After taking the decision it just moves to the next place in the selected direction.

When a Ghost-agent meets a Processing-agent it requests a simulated processing according to its internal state. It immediately receives the simulated new processing state, which it assumes as its new state. Then, the Ghost-agent moves on to the next place.

The run of the Ghost-agent is terminated at a place of an Unloader-agent. To reduce the computational load of the system, each Ghost-agent should be permitted only a small number of moves in the *PI* before it dies without having encountered an Unloader-agent.

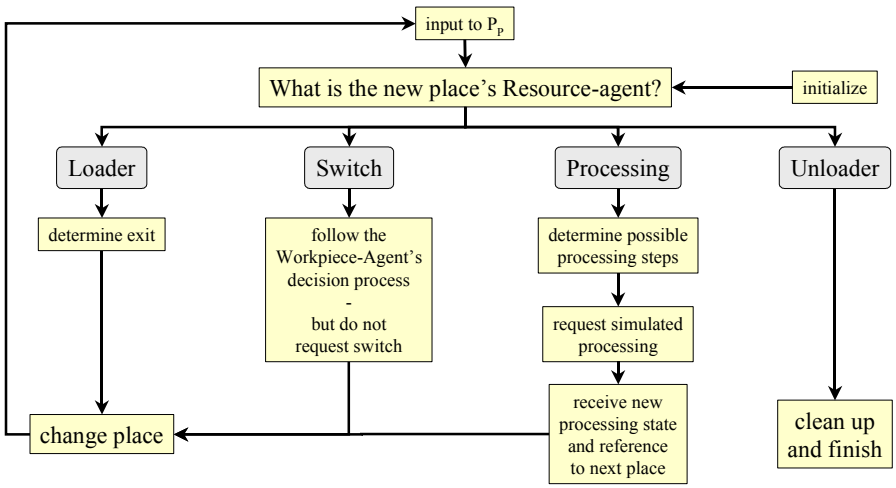


Figure 4.6. The Process Graph of the Ghost-agent

Figure 4.6 shows the process graph of a Ghost-agent. As a simulacrum of a Workpiece-agent, the graph of a Ghost-agent is very similar to that of a Workpiece-agent. The transport decision process of a Ghost-agent in the encounter with a Switch-agent is the same one its Workpiece-agent would follow. For more details on the actual decision see Figure 4.8.

Workpiece-agents jointly generate a representation of the current flow pattern by regularly refreshing $P_S$ pheromones according to their current processing state. Similarly, the interface layer defines a pheromone type "Prediction" ($P_P$), to which only Ghost-agents provide input. As $P_S$, $P_P$ has one additional data-slot "state", carrying an element from the finite set of processing states. The propagation parameter of $P_P$ is zero as well.

In contrast with a Workpiece-agent, a Ghost-agent never stays long enough at one place to generate an input with significant regularity. As a consequence, Ghost-agents only refresh a $P_P$ pheromone once at each place visited. The affected pheromone matches the internal state of the respective Ghost-agent. The strength of the input is specified in the global parameter "GhostRefreshStrength". Regularity of the refresh at one place is provided by the regular creation of Ghost-agents by the Workpiece-agents in the Monte Carlo simulation. The rate Ghost-agents are created with by a Workpiece-agent is set by the global parameter "GhostCreationRate". Workpiece-agents run an additional process in a closed loop, creating a Ghost-agent every "GhostCreationRate" units time.

The strength of a $P_P$ pheromone at a place is linked to the probable local load of workpieces in the state specified by the pheromone. The actual arrival of a workpiece at a zone depends on the outcome of processing steps and transport decisions yet to come.

As in Monte Carlo simulations in general, the quality of the prediction of the future pattern depends on the number of samples of the probabilistic model. In the case of the *GMC* system, the Ghost-agents running through the *PI*, emulating their respective Workpiece-agent, sample the model. The more often a Workpiece-agent generates a Ghost-agent, the better the quality of the prediction. The rate of the Ghost-agent generation must be tuned for the dynamics of the model change (changing yield, changing advice, etc.) and for the rate in which the advisory system accesses the prediction.

### 4.3.2 Integrating Advice

To integrate the advice given by the advisory system into the operation of the reactive layer of the control system the Switch-agents are enabled to delay the transport of a workpiece, and the decision processes of the Workpiece-agents and the Loader-agents are extended. These functional extensions are part of the specification of the interface layer.

#### Delaying the Transport

In contrast with the reactive layer, the interface layer permits Workpiece-agents to delay the further transport of their workpieces. Whereas the basic behavior of a Switch-agent demands a transport of workpieces even without a given direction, in its extended behavior a Switch-agent may be asked to delay a workpieces.

Depending on the layout of the elementary zone controlled by a Switch-agent, a transport of a workpiece is delayed by storing it in a local buffer, by routing it in an internal parking loop, or by simply stopping the workpiece where it is, blocking the flow.

No additional processes are added to the Switch-agent in the interface layer. The existing interaction and routing mechanisms are only extended by the delay option.

#### Guidance Pheromones

Advice is given through the pheromone type "Force" ($P_F$). The local strength of $P_F$ pheromones provides an additional bias to the probabilistic transport decisions of the Workpiece-agents and their Ghost-agents.

There is one additional data-slot to $P_F$. The slot "state" specifies a pheromone according to a processing state. In addition to all possible states, the slot may also take an "any state" value, indicating relevance independent of the processing state. An input to a $P_F$ pheromone propagates upstream with a non-zero propagation parameter.

The spatial strength pattern of $P_F$ pheromones represents the advice. Locally, the advice is meant to attract or to repel workpieces in their flow. Attraction is signaled in positive values of the strength of the $P_F$ pheromones. Repulsion uses negative strength values. The agents are designed to cope with a flow field of mixed negative and positive strength.

#### Extending the Workpiece-agent

The local strength of $P_F$ pheromones is incorporated into the routing decision process of a Workpiece-agent. As in its basic behavior, a Workpiece-agent first determines all possible next processing steps and samples $P_A$ to restrict the set of potentially available directions accordingly. But, in its extended behavior a Workpiece-agent then samples the current strength of all $P_F$ pheromones that match the processing state of its workpiece, including the pheromone "any state". It perceives the strength of the pheromones separately for each downstream direction (Section 3.3.1).

The final routing decision is taken in two steps that are illustrated in an example in Figure 4.7. First, a Workpiece-agent decides if it should request a fixed delay for its workpiece. If this is not the case, it selects a downstream direction according to the flow

field spanned by the $P_F$ pheromones sampled.

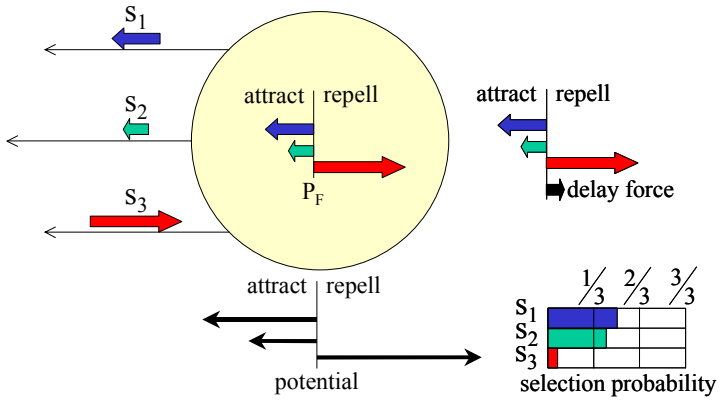


Figure 4.7. $P_F$ Biased Probabilistic Selection of the Exit Direction

The agent delays the transport based on a probabilistic decision. Let $s$ denote the sum of all previously sampled strength values of relevant $P_F$ pheromones. $s$ represents the general local downstream repulsion or attraction of the specific workpiece. The probability $P_d$ that the routing decision is delayed is computed using the following equation:

$$P_d = \begin{cases} 0 & : s \geq 0 \\ 1 - 2 * \left(1 + e^{-s * x_1}\right)^{-1} & : s < 0 \end{cases} \tag{4.1}$$

In the equation $x_1$ ($x_1 \geq 0$) is a constant parameter permitting the tuning of the decision process. Considering the definition of $P_d$, a workpiece is only delayed if its agent perceives a general downstream repulsion.

Finally, the Workpiece-agent picks a random number $rnd$ ($rnd \in [0,1]$). It requests a delay of the workpiece from the local Switch-agent, if $P_d \geq rnd$ holds. In this case, the Switch-agent delays the workpiece until further notice and the Workpiece-agent pauses for a fixed time before it restarts the whole decision process.

In the second stage of the extended routing decision, the Workpiece-agent finally selects the direction of the transport. If there is only one direction available, it simply requests the transport and eventually moves on. Otherwise, the Workpiece-agent sums up the previously sampled strength of the $P_F$ pheromones, but now specific for each permitted direction. Thereby, it perceives directed forces of repulsion or attraction.

The direction specific forces bias the otherwise random selection of a direction. The following requirements determine a force-biased selection out of a set of $n$ permitted directions:

1. A significantly stronger repulsion from one direction results in a low selection probability for this direction.
2. A significantly stronger attraction to one direction results in a high selection probability for this direction.
3. With no force from any direction, each exit is selected with a probability of *1/n*.

The following two-step algorithm determining the selection probabilities for each permitted direction meets these requirements. First, the force of all directions is used to compute a "potential" value for each direction, assuming that repulsion from one direction

is equivalent to attraction from all other directions and vice versa. Hence the potential value $pot_i$ of $n$ directions is computed as:

$$pot_i = 2 * s_i - s, i = 1..n \tag{4.2}$$

The value $s_i$ is the sum of all $P_F$ pheromones relevant for direction $i$ and $s$ is computed as before as the sum of all relevant $P_F$ pheromones. With all forces localized, in a second step the selection probability $P_i$ for each direction is computed by:

$$P_i = 1 - \frac{1}{1 + e^{\ln(n-1) - pot_i}}, i = 1..n \tag{4.3}$$

These probability values fulfill the specified requirements. Most importantly, they provide a random selection of a direction in the absence of any advice from the advisory system. As long as the advisory system is not active, the control system operates in the extended behavior of the Workpiece-agents as it does in the basic behavior specified in the reactive layer.

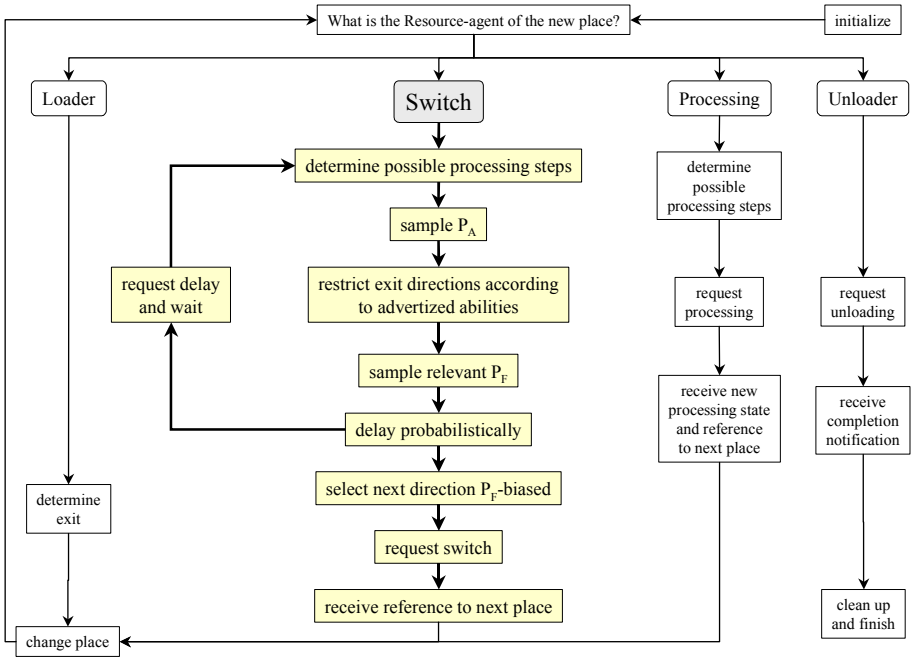


Figure 4.8. Extended Behavior of the Workpiece-agent

The selected direction enables the Workpiece-agent to request from the Switch-agent a transport of its workpiece. In Figure 4.8, the extended behavior of the Workpiece-agent is shown, focusing on the new transport decision process.

### Guiding the Inflow

The extension of the behavior of a Loader-agent is similar to the extended Workpiece-agent behavior where a product specific delay of the material flow is the control instrument.

In its basic behavior a Loader-agent implements a given product mix and a preset rate when loading workpieces into the manufacturing system. In every cycle it first selects the next product according to the required mix. Then, the agent tries to load a workpiece

matching the selected product.

In its extended behavior the Loader-agent also selects a product first. But, then it samples the current strength of $P_F$ pheromones. Since workpieces of different products may have different initial states, the Loader-agent samples only pheromones that are relevant to the potential workpiece.

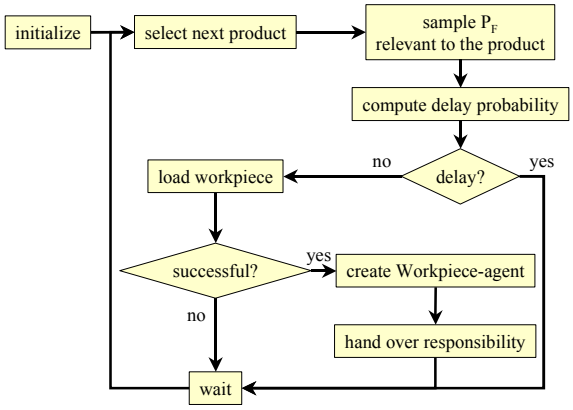


Figure 4.9. $P_F$ Biased Probabilistic Loading

The potential workpiece has no agent yet. The Loader-agent carries the responsibility for the workpiece, deciding if it should be delayed or if it is loaded into the system. The decision process is based on the same algorithm a Workpiece-agent would use (Equation (4.1)). The Loader-agent sums up all relevant $P_F$ pheromones and computes a probability $P_d$. On the basis of $P_d$, the loading of the workpiece is canceled or it commences as in the basic behavior.

Figure 4.9 depicts the process graph of the extended loading cycle.

## 4.4 Advisory System: Strategy Implementation Layer

The advisory system is the second major component of the *GMC* system. The following presentation focuses on the bottommost layer of the advisory system, the strategy implementation layer. The task of the strategy implementation layer is to generate advice for the underlying control system. It operates on the material flow patterns provided by the interface layer, accessing an aggregated current state of the manufacturing system and a prediction of the near future.

The concept of a (material flow) "strategy" is the central element of the advisory system. It represents a flow pattern that is to be achieved. The strategy implementation layer is given such a strategy for implementation. The advice generated by the layer attempts to guide the control system towards the goal pattern. The agents of the layer try to achieve and maintain the current strategy – a flow pattern in the manufacturing system that mirrors the current goal pattern.

Structurally, a goal pattern matches the current or the predicted pattern generated by the interface layer. Local load values are given for specific to processing states. A strategy may define a local load for all or for only some places, permitting the strategy implementation layer to act in the *PI*. The advice is generated distributedly, expressed in the $P_F$ pheromones (Section 4.3).

In the strategy implementation layer a new agent type is introduced. Operating solely in the virtual environment of the *PI*, Policy-agents attempt a distributed implementation of the current strategy. Each Policy-agent represents the local load that is to be achieved.

The semantics of advice depends on the local context in the manufacturing system. The advisory system specifies a set of propagation filters in the *PI* (Section 3.3.1) that provide an automatic translation of advice. The chosen design demonstrates the ability of the *PI* to actively process information, thereby reducing the computational complexity of the agents.

### 4.4.1 The Policy-Agent Type

The current strategy is represented and implemented by a set of Policy-agents. There is one Policy-agent at most for each place in the *PI*. If there is a place without a Policy-agent, the strategy is not defined for the place and any material flow pattern is accepted there. Figure 4.10 illustrates the structure of a material flow strategy and its mapping to Policy-agents.

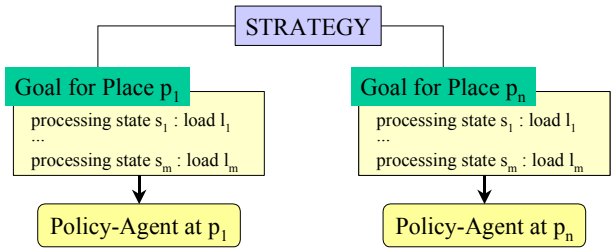


Figure 4.10. Distributed Representation of a Material Flow Strategy

Each Policy-agent knows the processing state specific load values required for its place, and it has access to the local $P_S$ and $P_P$ pheromones representing the local aspect of the current and the predicted material flow pattern. The goal of a Policy-agent is to match the current load values with the required ones.

The implementation of the current strategy by Policy-agents may be realized on different levels of sophistication, using reasoning ranging from simple to complex. Here the *GMC* system realizes a simple difference-driven approach.

The activities of a Policy-agent take the current and the predicted material flow into account. The load values of the current pattern are based on workpieces that are already in the zone or that even have left it. There is nothing a Policy-agent can do to change the behavior of the agents of these workpieces. On the other hand, the prediction of the flow is based on an approximation of the behavior of Workpiece-agents still located upstream. The Policy-agents adjust the behavior of the Ghost-agents, which mirror the routing decision process of their Workpiece-agents. Based on repeated Monte-Carlo simulations, the Ghost-agents and the Policy-agents negotiate the best configuration of the virtual environment, in which the Workpiece-agents later take their decisions.

A Policy-agent operates on the absolute load values given by the current load pattern and on relative load values predicted by the near-future pattern. The emergent prediction of a flow pattern is accurate in representing the expected relative load values in the relative strength of the respective $P_P$ pheromones. The expected absolute load is not extractable from the sum of the $P_P$ pheromones. There are too many uncertainties and abstractions in the prediction process. On the other hand, the representation of the current absolute load in the $P_S$ pheromones is very accurate, as demonstrated in Section 4.5.

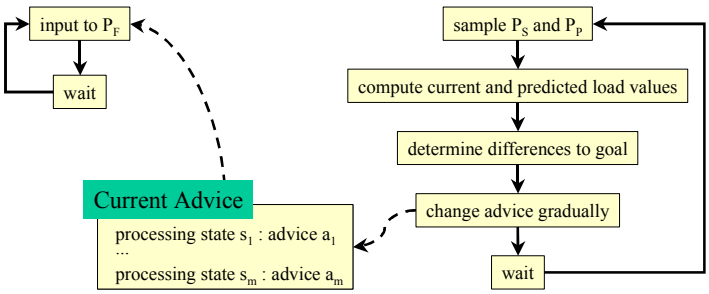


Figure 4.11. Two Processes of a Policy-agent

Figure 4.11 illustrates the two processes executed by a Policy-agent. One process transmits the current advice by regularly refreshing local $P_F$ pheromones according to its current intended attraction or repulsion of workpieces in specific processing states. The second process re-evaluates the advice. As the transmission of the advice, the adaptation of the advice runs in a closed loop. Starting a new adaptation cycle, the Policy-agent samples all available $P_S$ and $P_P$ pheromones first. Assuming the strength of these pheromones to be in equilibrium, it then extracts the processing state specific local load values.

With all external data available, the agent determines its next activities. In a first step, it normalizes the predicted load values. Thus all normalized load values of the predicted pattern sum up to one. Then, for each processing state, the normalized predicted load value is subtracted from the normalized load value of the goal of the Policy-agent. Thus, the required increase in share (positive subtraction result) or decrease in share (negative subtraction result) is computed. But, an increase of one relative load may be achieved by increasing the respective load or by decreasing all the others, and vice versa the decrease.

To decide how the change in the shares is achieved, the Policy-agent consults the absolute current load. If the absolute current load is smaller than the absolute load goal, the current load must increase. Thus, the load of workpieces in underrepresented processing states should be increased. If the absolute load has to decrease – a decrease of specific load values is required.

If there is no change of any processing state specific relative load value required but the absolute load is not yet as specified in the goal of the Policy-agent, the advice must be adapted for all processing states equally.

As a result of its reasoning process, the Policy-agent knows which workpieces to attract or to repel stronger than before. According to SE design principle 14, gradual change in the behavior of an agent is to be preferred to abrupt changes. Thus, a Policy-agent changes its transmitted advice only by a small and fixed amount in the adaptation process, even if the current and predicted patterns differ strongly from the given goal pattern.

After having adapted its advice, the Policy-agent has completed another adaptation cycle. The process is paused for a fixed time and then the next adaptation cycle begins. The behavior of a Policy-agent is parameterized by the refresh rate of the $P_F$ pheromones and the rate and strength of the change in one cycle of the adaptive process.

### 4.4.2 Context Dependence

Advice given by Policy-agents depends on the local context of the manufacturing system. Spatial propagation of input events to $P_F$ might leave the local context the advice

is given for. The local context of an elementary zone in the manufacturing system is set by the processing state of the workpieces passing through it. For advice to be effective, it must be stated in relation to these workpieces.

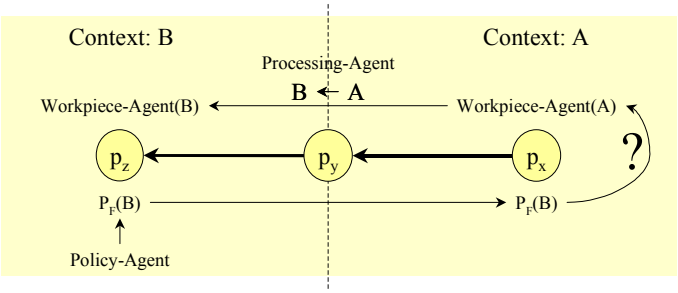


Figure 4.12. Local Context in the Manufacturing System

The following example (Figure 4.12) illustrates the context dependence of advice. Assume that there is a sequence of three places in the *PI* ($p_x$, $p_y$, $p_z$) with the material flow running from $p_x$ to $p_y$ and on to $p_z$. Further assume that there is a Processing-agent at $p_y$ and Switch-agents at $p_x$ and $p_z$. The only processing capability available at $p_y$ is to change the processing state of a workpiece from *A* to *B*. Thus, only workpieces in state *A* pass $p_x$, whereas at $p_z$ there are only workpieces in state *B*. Then, the local context at $p_x$ is *A*, and at $p_z$ it is *B*.

A Policy-agent located at $p_z$ perceives a stream of workpieces in state *B*. Therefore, its advice is given in the $P_F$ pheromone *state=B*. Inputs to this pheromone are propagated upstream. $p_x$ is the next place where Workpiece-agents access the advice. But, since they are all in state *A*, they never sample the $P_F$ pheromone *state=B* because it is not relevant to them. At $p_y$ the context changes because there the processing state is changed.

It is necessary to translate the advice according to the context of the respective zone. The translation takes place during the propagation of input events, realized in a specific propagation filter in the *PI* (Section 3.3.1).

To translate the advice expressed by $P_F$, a change in the context must be accessible to the *PI*. A change of context occurs only at places of Processing-agents. The change is characterized by the transitions of the processing state of workpieces caused by the agent. Therefore, for every state transition occurring, a Processing-agent generates an input to a pheromone of the "Transition" ($P_T$) type.

$P_T$ has two additional data slots, both carrying a processing state. The first slot ("in") contains the processing state, with which a workpiece has arrived, whereas the second one ("out") takes the processing state, with which the workpiece has left. Inputs to $P_T$ do not propagate and it evaporates extremely slowly. Over a long period, the relative strength of the $P_T$ pheromones approximates the long-term yield of the processing unit.

A propagation filter, translating input events to $P_F$ during their upstream propagation, is only active where $P_T$ pheromones are available. At such a place, the propagation of an input to $P_F$ is blocked if the content of the "state" slot of the respective $P_F$ pheromone matches the "out" slot of a local $P_T$ pheromone. Instead, a "virtual" input to another $P_F$ pheromone is propagated further upstream. The input refers to a $P_F$ pheromone that is specified by the value of the "in" slot of the matching $P_T$ pheromone. The propagation of an input to $P_F$ is also changed quantitatively. The strength of the translated advice is based on the inverse of the yield, as it is perceived in the relative strength of the matching $P_T$ pheromone.

More complex translations occur if there is more than one processing state permitted at the entrance of a processing unit. Assume that workpieces arrive in state *A* or *B*, and they leave in state *C* or *D*. All four possible transitions actually occur. In this case, advice relevant for *D* is translated to advice for *A* and for *B*.

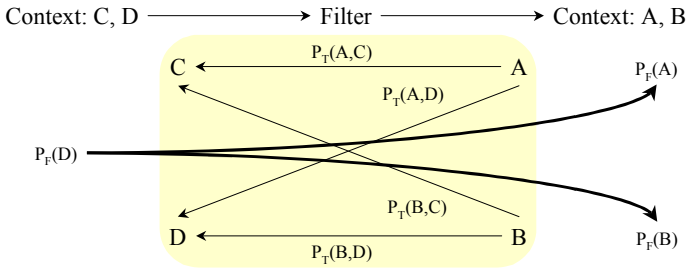


Figure 4.13. Example of Context-Filtering

The translation is realized by splitting the strength of the original advice according to the relative strength of the $A \rightarrow D$ and the $B \rightarrow D$ transitions ($P_T$ pheromones) first. Then, the resulting strength is divided by the relative strength of the $A \rightarrow D$ and $A \rightarrow C$, or the $B \rightarrow D$ and $B \rightarrow C$ transition respectively (Figure 4.13).

The following equations specify the required computation. The terms *F(A)* and *F(B)* denote the strength of the resulting advice relevant to the respective processing states ($P_F$*(state=A)*, $P_F$*(state=B)*). Similarly, *T(x,y)* denotes the strength of a $P_T$ pheromone with "in" set to *x* and "out" set to *y*.

- $F(A) = F(D)\dfrac{T(A,D)}{T(A,D)+T(B,D)}\dfrac{T(A,D)+T(A,C)}{T(A,D)}$
- $F(B) = F(D)\dfrac{T(B,D)}{T(A,D)+T(B,D)}\dfrac{T(B,D)+T(B,C)}{T(B,D)}$

With the $P_T$ pheromones in place the Ghost-agents may not have to interact with the Processing-agents directly. Instead they may determine the outcome of a simulated processing themselves, based on the yield represented in these pheromones.

## 4.5 Demonstration and Evaluation

Based on the specification presented in the previous sections, the operation of the *GMC* system is discussed in a small but realistic demonstration. The manufacturing system is abstracted from the paint shop problem (Section 4.4.1) with adaptations for demonstration purposes.

Traversing the system architecture from bottom up, the structure of the *PI* is derived from the assumed manufacturing system first. Then, in the following sections, the operation of the reactive layer, the interface layer, and finally the strategy implementation layer is discussed and evaluated using the results gained in the analysis of the formal model of the *PI*.

In the evaluation of the layers, the numerically computed predictions of the resulting global behavior and of the pheromone patterns are compared with observations in a prototypical implementation of the *GMC* system.

Besides evaluating the system at hand, the demonstration also serves as an example for the general approach to tuning and evaluation of multi-agent coordination on the basis of the *PI*.

### 4.5.1 From Reality to the PI

Figure 4.14 depicts the *PI* as it is described in the following discussion. Each icon of a place shows the task fulfilled in the respective zone (Load, Switch, Process, Unload), and the time it takes a workpiece take to pass the zone. The places of the processing units also display the yield of the respective unit.

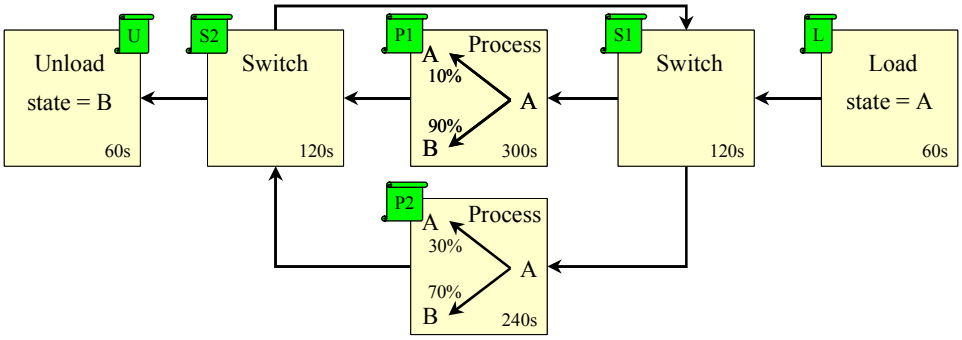


Figure 4.14. The PI in the Demonstration

There is only one processing step executed in the manufacturing system. The workpieces enter it in a processing state “A” and they are permitted to leave only if they are in state “B”. Segments of the paint shop application matching this more abstract demonstration are repair units like “Spot-Repair” or “Repair-Preparation”.

There are two processing units (“P1”, “P2”). Both accept “A”-state workpieces and out of both units emerge workpieces that are in state “A” when no processing happened. Workpieces leave the units in state “B” if the processing was successful.

Each processing unit has a yield (“A” vs. “B”) and specific processing times. After passing unit “P1”, an average of 90% of the processed workpieces is in state “B” and only 10% remain in state “A”, while for unit “P2” the yield is only 70% state “B” versus 30% state “A”. But, on the other hand, unit “P1” takes longer (300 seconds) to process a workpiece than unit “P2” (240 seconds).

The workpieces all enter the manufacturing system through one entry, and they all leave through the same exit. The average loading rate is assumed to be one workpiece every 60 seconds.

The flow of workpieces coming from the entry is split in front of the processing units. After processing, the flows join to be split up again according to the processing state of the workpieces. All workpieces in state “B” have to go to the exit. All remaining state “A” workpieces return to the zone in front of the processing units.

Following the given restrictions for allocating elementary zones, the manufacturing system is divided into six zones. There is a zone for each processing unit. Additionally, there is a small zone for the entry and one for the exit. Finally, an elementary zone between the entry and the processing units and one after the processing units is identified.

“L” denotes the entrance zone and it takes a workpiece 60 seconds to pass it. It then

moves into the zone where the decision for one of the processing units is taken (“S1”). The workpiece emerges at the entrance of the respective zone of one of the processing units after 120 seconds.

After passing the processing units in either zone “P1” or “P2”, the flow of workpieces merges again in zone “S2”. 120 seconds later they leave “S2” again to enter “U”, the zone of the exit, or “S1” according to the respective processing state. Finally, unloading is completed after 60 seconds.

Each elementary zone in the manufacturing system is mapped to a separate place in the *PI*. The places are linked according to the layout of the transport system.

Each zone is given one Resource-agent according to the local activities. A Loader-agent is created for the entrance zone “L”. “S1” and “S2” are controlled by Switch-agents, and “P1” and “P2” are covered by Processing-agents. The exit zone “U” is assigned an Unloader-agent. For simplicity, the places and the agents too are named after their respective elementary zone.

### 4.5.2 Reactive Layer Operation

The reactive layer is made up of Resource-agents and Workpiece-agents. The Resource-agents are already placed in the *PI* as specified in the previous section. A Workpiece-agent is created by the single Loader-agent whenever a workpiece is successfully loaded into the manufacturing system.

All Workpiece-agents carry the processing state of their workpiece. Initially, the state is “A”. It is eventually changed to “B” during processing. Only “B”-state workpieces are permitted to proceed to zone “U”.

The central coordination mechanism of the reactive layer is the attraction of workpieces to the processing units and later to the exit. The attraction is communicated in two $P_A$ pheromones. The first pheromone (ability=“A2B”) represents the capability of the two processing units to take in workpieces of state “A” and process them into state “B”. The Unloader-agent, on the other hand, refreshes the standard $P_A$ pheromone (ability=“unloading”).

Each Processing-agent (“P1”, “P2”) regularly refreshes the $P_A$ pheromone “A2B” whenever the processing ability is available. The refresh rate and the refresh strength are specified by the two global parameters “AbilityRefreshRate” and “AbilityRefreshStrength”. The emerging spatial strength pattern depends on the evaporation parameter and on the propagation parameter of $P_A$.

An input to $P_A$ propagates upstream to the flow of workpieces. Thereby, places upstream from the respective processing resource receive routing information.

In Figure 4.15 the $P_A$ pattern emerging for a specific set of parameters is shown. The parameters “AbilityRefreshRate” and “AbilityRefreshStrength” are both set to a value of one. The evaporation parameter is set to 0.99 (percent remaining after one second), indicating a slow evaporation, and a propagation parameter of 0.5 (percent remaining after one propagation step) resulting in a small attractive radius.

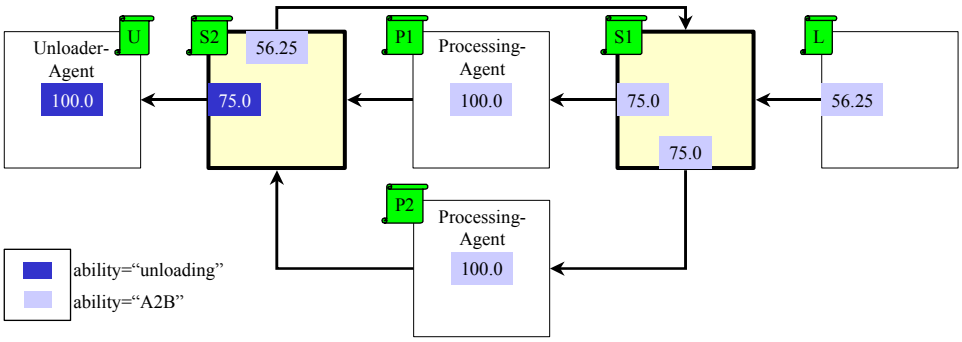


Figure 4.15. Spatial $P_A$ Pattern

At place "S1" the Workpiece-agents all carry state "A". Hence, they need to know, which direction to take to a processing unit that provides the ability "A2B". The upstream propagation of input events from the Processing-agents and the direction-specific aggregation of $P_A$ pheromones at "S1" point Workpiece-agents to downstream directions where the required ability is available. If both processing units are online, the selection is taken randomly.

The next routing decision is required at "S2". The regular refresh of the $P_A$ pheromone ability="unloading" by the Unloader-agent provides Workpiece-agents in state "B" with guidance. But, if a workpiece is still in state "A", the agent is led by the two-step upstream propagation ("P1"/"P2" to "S1" to "S2") of the processing ability "A2B".

Workpiece-agents are created at place "L". They pass into "S1" when their workpiece is loaded. At "S1" they select randomly "P1" or "P2". The selection results in a transport request to the Switch-agent. The workpiece is sent to the zone of one of the processing units.

After the transport is completed by the Switch-agent, the Workpiece-agent passes on to the next place. There it encounters a Processing-agent ("P1" or "P2") and requests the processing of its workpiece using the capability "A2B". The Processing-agent realizes the processing and tells the Workpiece-agent the new state of its workpiece. The new state of the workpiece depends on the yield of the processing unit. "P1" offers a ninety percent chance for a successful processing (result: "B"). At "P2" the chance is only seventy percent. After the processing result is transmitted, the Workpiece-agent moves to the next downstream place, which is "S2" in the demonstration.

At "S2" the transport decision is based on the state of the workpiece. Workpieces in state "B" enter the zone "U" and their Workpiece-agents meet the Unloader-agent. Whereas, workpieces in state "A" return to zone "S1", taking their agents to meet Switch-agent "S1" again.

At place "U" a Workpiece-agent requests the unloading of its workpiece. After completion, the Workpiece-agents are notified, telling them that their task is fulfilled and they are permitted to die.

### 4.5.3 Pattern Generation in the Interface Layer

Two patterns are generated to provide the advisory system with information. The spatial pattern of $P_S$ pheromones represents the current material flow whereas $P_P$ pheromones provide a prediction of the near future.

The $P_S$ pattern is generated by the Workpiece-agents, refreshing the pheromone that matches the current processing state of the workpiece. All Workpiece-agents show such a refresh behavior at all places. It is specified by the global parameters "StateRefreshRate" and "StateRefreshStrength".

Figure 4.16 shows the pattern generated in the manufacturing system of the demonstration. The global refresh parameters are set to "StateRefreshRate"=10 seconds and "StateRefreshStrength"=1. The evaporation parameter is set to 0.99 (percent remaining after one second). Simulating the manufacturing system of the demonstration while running the agents generates the data. But, it may be predicted analytically too, as the following discussion shows.

Workpiece-agents enter the system at a rate of 1 agent per 60 seconds. Assuming complete availability of the processing units, the flow of workpieces is split evenly at "S1". Hence statistically, every two minutes an agent arrives at "P1" as well as at "P2".

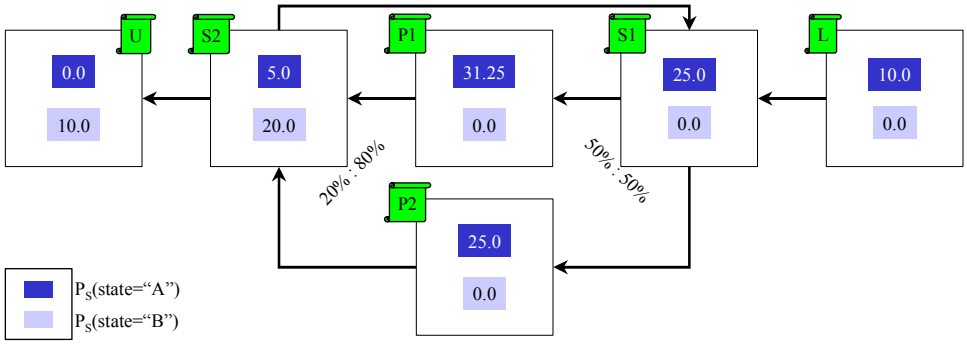


Figure 4.16. Simulated $P_S$ Pattern

For each processing unit there is a fixed yield. At "P1" ninety percent of all arriving workpieces leaves in state "B", and only ten percent leave in state "A". Hence, the flow of "A"-state workpieces out of "P1" is 10 percent the strength of the inflow.

Because of the fifty-fifty split of the flow between the two processing units, and because of their processing yield, the inflow of Workpiece-agents into place "S2" is 80 percent "B"-state and 20 percent "A"-state workpieces. The strength of the inflow is the same as the outflow strength of "S1".

At this stage, there are 0.8 Workpiece-agents in state "B", entering "S2" every minute. "A"-state agents arrive in a rate of 0.2 agents per minute. They continue further to "S1" where they enter the flow of one agent per minute generated by the Loader-agent. Hence, the output of "S1" adds up to 1.2 agents per minute split evenly between the processing units.

Of the 1.2 agents per minute, twenty percent return to "S1" eventually. Repeatedly adding the fed back flow yields the following term for the rate of Workpiece-agents passing "S1": $\sum_{i=0}^{\infty} 0.2^i = \frac{1}{1-0.2} = 1.25$.

Knowing the absolute flow at "S1", the rest of the pattern of flow rates may be computed. The following table lists the computed flow rates for all places (in agents/minute) specific to the processing states:

Table 4.1. The Current Pattern of Load-Values.

| | "L" | "S1" | "P1" | "P2" | "S2" | "U" |
|---|---|---|---|---|---|---|
| inflow "A" | — | 1.25 | 0.625 | 0.625 | 0.25 | 0.0 |
| inflow "B" | — | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 |
| outflow "A" | 1.0 | 1.25 | 0.0625 | 0.1875 | 0.25 | — |
| outflow "B" | 0.0 | 0.0 | 0.5625 | 0.4375 | 1.0 | — |

From the strength of the flow and the time an agent is present at a place, the number of agents statistically present at a place at any time is computed by simple multiplication. With the number of agents known, the pheromone equilibrium is computed for each place and each processing state (Equation 3.26). As Table 4.2 shows, the predicted strength of $P_S$ pheromones in equilibrium is very closely approximated by the respective average strength in Figure 4.16.

Table 4.2. Prediction of the Pattern of $P_S$ Pheromones.

| | "L" | "S1" | "P1" | "P2" | "S2" | "U" |
|---|---|---|---|---|---|---|
| $P_S$(state="A") | 10.0 | 25.0 | 31.25 | 25.0 | 5.0 | 0.0 |
| $P_S$(state="B") | 0.0 | 0.0 | 0.0 | 0.0 | 20.0 | 10.0 |

The deviation of the actual pheromone strength from the computed prediction results from the non-deterministic routing behavior at "S1", introducing perturbations in the strength values. The visibility of a perturbation depends on its strength, its duration, and on the evaporation parameter. Slower evaporation in the same scenario results in less fluctuation, as illustrated in Figure 4.17 where the strength of $P_S$ at "P1" is shown in the same scenario for different evaporation parameters.

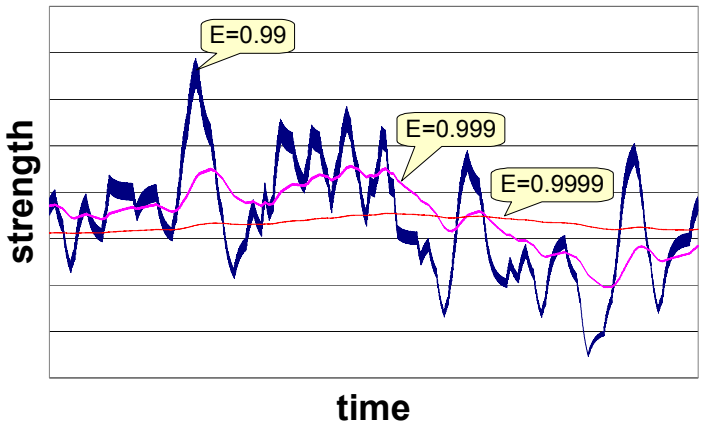


Figure 4.17. $P_S$ at "P1" with Different Evaporation Parameters (E)

The analytic prediction of the average pheromone strength also provides the argument for the perception mechanism of the Policy-agents of the strategy implementation layer. Knowing the fixed refresh behavior of a single Workpiece-agent, the parameters of $P_S$, and the average transport or processing times, the local load may be perceived in the $P_S$ pattern. The evaporation parameter determines how up-to-date the aggregated information is.

The pattern generated by the interface layer in the $P_P$ pheromones represents a prediction of the relative load in the material flow in the near future. Like $P_S$, processing state information specifies $P_P$ pheromones.

The mechanism that anticipates the near future simulates the material flow. Each Ghost-agent emulates the behavior of the Workpiece-agent by which it was created. It takes transport decisions based on the same information and using the same algorithm. But, since a Ghost-agent does not accompany a workpiece, it does not have to wait at each place for the handling of the workpiece to be completed. Instead, it just moves on to the next place at "electronic" speed.

Processing steps need to be simulated during the run of a Ghost-agent. Processing-agents keep a floating average of the yield of their processing unit. On request they determine a result of a simulated processing step according to the probabilities given by the yield. Passing Ghost-agents make the request and its outcome changes their internal state.

A Ghost-agent refreshes the $P_P$ pheromone matching its current internal state. There is only one refresh after every move from one place to another.

The prediction process is influenced by similar parameters as the generation of the current pattern. To generate a regular refresh of $P_P$, Workpiece-agents regularly create Ghost-agents at a fixed rate specified by the global parameter "GhostCreationRate". "GhostRefreshStrength" sets the refresh strength of Ghost-agents. An additional parameter influencing the accurateness of the prediction is the number of moves, each Ghost-agent is permitted to make in the simulation. This parameter determines how far into the future the behavior of a Workpiece-agent is emulated.

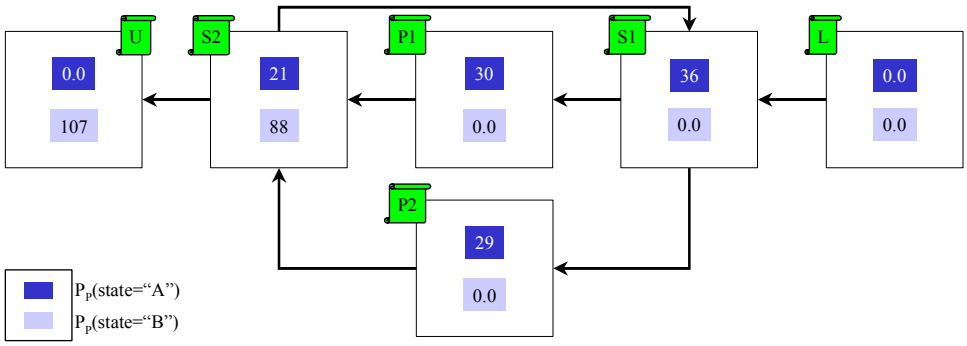


Figure 4.18. Simulated $P_P$ Pattern

Figure 4.18 shows the $P_P$ pattern emerging in the simulation. The "GhostCreationRate" of Workpiece-agents is set to one Ghost-agent every 10 seconds. The "GhostRefreshStrength" has a value of one. Each Ghost-agent is permitted to make ten moves at most. Again, the evaporation of $P_P$ pheromones is set to 0.99 (percent remaining after one second).

Based on the average load values of each place and the even split of the material flow at "S1", the average $P_P$ pattern may be predicted analytically too. The following argument presents the approach.

There are on average 3.125 Workpiece-agents at place "P1". Together, they create 18.75 Ghost-agents per minute. Taking the simulated processing at "P1" into account, 1.875 Ghost-agents leave "P1" in state "A" while 16.875 agents emerge in state "B" every minute. The input to $P_P$ starts when entering "S2", because this is the first move of the Ghost-agents that are created at "P1".

Ghost-agents in state “B” move on from “S2” to place “U”, whereas “A”-state agents continue to “S1”. At place “U” they encounter the Unloader-agent and die. At “S1” they emulate the transport decision of Workpiece-agents and split evenly between “P1” and “P2”. Thus 0.9375 Ghost-agents re-enter “P1” per minute.

With a maximum of ten moves, the longest run of Ghost-agents created at “P1” is: “P1”-”S2”-”S1”-”P1”/”P2”-”S2”-”S1”-”P1”/”P2”-”S2”-”S1”-”P1”/”P2”-”S2”. The path of Ghost-agents has to be followed to describe the complete flow pattern emitting from “P1”. The overall flow pattern of Ghost-agents in the demonstration is calculated by tracking the Ghost-agents that are generated at each place. The volumes of the flows add up since there are no dependencies among Ghost-agents. The following table depicts the resulting relevant load rates at each place:

Table 4.3. The Predicted Pattern of Load-Values.

| | “L” | “S1” | “P1” | “P2” | “S2” | “U” |
|---|---|---|---|---|---|---|
| Load “A” | 0.0 | 22.857 | 18.8925 | 18.8925 | 13.908 | 0.0 |
| Load “B” | 0.0 | 0.0 | 0.0 | 0.0 | 57.507 | 69.303 |

Each Ghost-agent refreshes a $P_P$ pheromone once when it enters a new place. Hence, the global flow rates are equal to the resulting average refresh rates of $P_P$. With the refresh strength and the evaporation parameter given, the equilibrium values are computed as listed in the following table:

Table 4.4. Prediction of the Pattern of $P_P$ Pheromones.

| | “L” | “S1” | “P1” | “P2” | “S2” | “U” |
|---|---|---|---|---|---|---|
| $P_P$(state=“A”) | 0.0 | 38 | 31 | 31 | 23 | 0.0 |
| $P_P$(state=“B”) | 0.0 | 0.0 | 0.0 | 0.0 | 96 | 116 |

The discussion also shows that the resulting equilibrium values of the $P_P$ pheromones depend strongly on the parameters of the simulation, especially on the permitted number of moves. Hence, the absolute strength values cannot provide a load prediction in the same way the strength of the $P_S$ pheromones does for the current pattern. But the relative strength in the mix provides useful information. The anticipated load split between “P1” and “P2” is represented in the relative strength of $P_P$ pheromones at “P1” and “P2” (fifty-fifty), while the predicted combined yield of “P1” and “P2” with 20 percent state “A” and 80 percent state “B” is given in the pheromone mix (Section 3.2.4) at “S2”.

### 4.5.4 Heeding Advice

Advice is transmitted in the local strength of $P_F$ specific for each processing state. Workpiece-agents heed the advice when taking transport decisions. The strength of the relevant $P_F$ pheromones matching the processing state of the respective workpiece is sampled. These values are then used to determine the probability for delaying the workpiece (Equation 4.1) and they bias the otherwise random selection of one of the permitted directions.

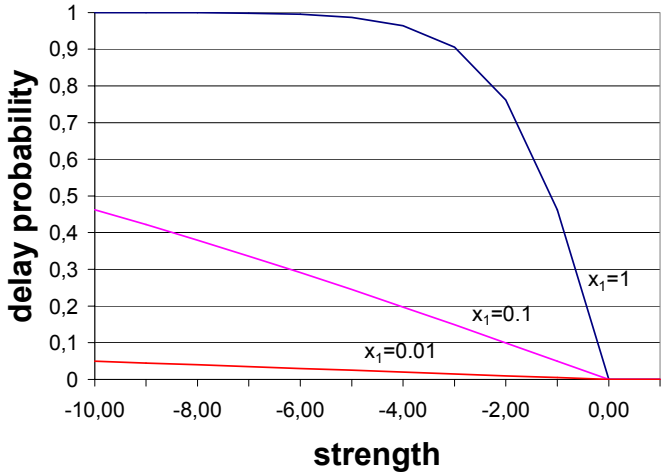


Figure 4.19. Probability Function for Workpiece-Delay

The delay probability $P_d$ is used by Workpiece-agents and also by the Loader-agent to determine the probability for a delay of a transport decision for a fixed time. Figure 4.19 plots the function for specific parameter values. It is the parameter $x_1$ that determines the steepness of the function for negative strength values. The larger $x_1$ the weaker the $P_F$ pheromones may be to result in the same delay probability.

If the workpiece is to be routed, the Workpiece-agent selects the exit to which the workpiece should be taken. In the extended Workpiece-agent behavior specified in the interface layer, the selection is based on $P_A$ and on $P_F$. $P_A$ pheromones reduce the set of generally available directions to exits that offer processing of the next required step – a hard constraint already fulfilled by the reactive layer.

In the reactive layer the Workpiece-agent selects a direction randomly. In the presence of $P_F$, the selection is biased. The strength of the relevant pheromones is interpreted as attracting or repelling force specific for each exit. Positive strength attracts, whereas a negative one repels workpieces. The selection probability for each permitted direction is computed in a two-step procedure. First, all strength values for all exits are aggregated into "potential" values where attraction towards one exit is interpreted as repulsion from all other directions.

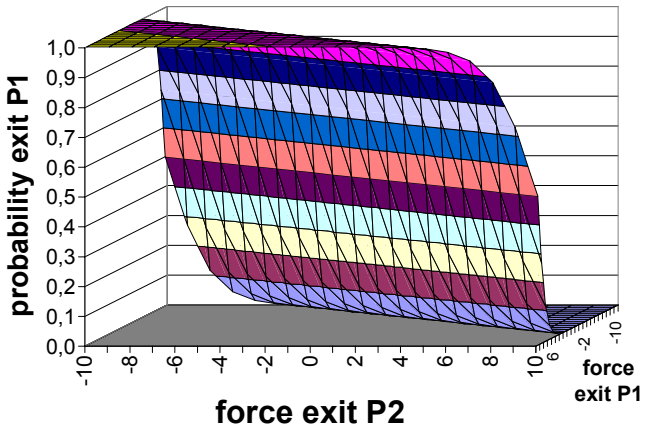


Figure 4.20. Probability Function for Exit-Selection

In the second step, the "potential" values are used to compute the selection probabilities. The probability values of all permitted exits add up to one. In the absence of any advice, each exit has the same probability of being selected, automatically returning the system to the basic behavior of the reactive layer. Figure 4.20 plots the selection probability function for one direction in a two-options choice.

In the demonstration, there is only one place where Workpiece-agents may have a two-exit choice. At "S1" they may select the exit to "P1" or the one to "P2". As the discussion of the mechanisms in the reactive layer already shows, the flow pattern emerging from the transport decision and the processing yield is numerically predictable.

Table 4.5. The Predicted (Current) Pattern for a 66/34 Split.

| | "L" | "S1" | "P1" | "P2" | "S2" | "U" |
|---|---|---|---|---|---|---|
| inflow "A" | — | 1.2 | 0.8 | 0.4 | 0.2 | 0.0 |
| inflow "B" | — | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 |
| outflow "A" | 1.0 | 1.0 | 0.08 | 0.12 | 0.2 | — |
| outflow "B" | 0.0 | 0.0 | 0.72 | 0.28 | 1.0 | — |
| $P_S$(state="A") | 10.0 | 24.0 | 40.0 | 16.0 | 4.0 | 0.0 |
| $P_S$(state="B") | 0.0 | 0.0 | 0.0 | 0.0 | 20.0 | 10.0 |

Table 4.5 lists the expected flow and the resulting $P_S$ strength values. Assume a 66/34 probability for the selection of "P1". Out of three workpieces arriving at "S1" two are processed at "P1" and one goes to "P2". Positive $P_F$ strength is assumed, excluding any delay options from the prediction.

Table 4.6. The Predicted (Predictive) Pattern for a 66/34 Split.

| | "L" | "S1" | "P1" | "P2" | "S2" | "U" |
|---|---|---|---|---|---|---|
| Load "A" | 0.0 | 19.3627 | 22.2571 | 11.4658 | 10.9880 | 0.0 |
| Load "B" | 0.0 | 0.0 | 0.0 | 0.0 | 56.3266 | 68.2013 |
| $P_P$(state="A") | 0.0 | 32 | 37 | 19 | 18 | 0.0 |
| $P_P$(state="B") | 0.0 | 0.0 | 0.0 | 0.0 | 94 | 114 |

This prediction is validated in simulations where the advice is transmitted by an agent refreshing the $P_F$ pheromone state="A" at "P1" regularly. The refresh strength is set to 7.0 and the rate is one input per second. Section 4.5.5 discusses the resulting $P_F$ pattern.

Ghost-agents, which are regularly created by Workpiece-agents, emulate the behavior of their creator in the near future. Ghost-agents heed the advice given via $P_F$. Table 4.6 lists the prediction of the flow of the Ghost-agents and the resulting $P_P$ pattern in the demonstration under a 66/34-split advice at "S1". Again, the "GhostCreationRate" parameter is set to ten.

### 4.5.5 Strategy Implementation

Implementing a strategy means influencing the material flow to reach and maintain a specified goal pattern. A strategy defines a goal pattern in terms of local flow requirements. For any elementary zone in the manufacturing system, goal values for the load and the mix of the local material flow may be stated.

A Policy-agent is located at a place in the *PI* and it tries to fulfill the given local goal. The agent runs two processes. One process transmits the advice, while the other one adapts it.

In the process of transmitting advice, $P_F$ pheromones are regularly refreshed according to the current advice of the Policy-agent. The input events are propagated upstream the material flow, influencing transport decisions of Workpiece-agents and Ghost-agents. The refresh behavior is specified by internal parameters defining the current refresh strength for each $P_F$ pheromone.

The adaptive process changes the parameters of the refresh behavior of a Policy-agent. The adaptation is guided by the difference between the current and predicted situation in the manufacturing system and the goal of the Policy-agent. Regularly, the strength of $P_S$ and $P_P$ pheromones is sampled and the current load and mix as well as the predicted mix are retrieved. Comparing these values to the goal of the agent, small changes in the advice might be deduced.

Assume that the current strategy expects a load of 4.0 workpieces in state "A" at "P1". There is a Policy-agent located at place "P1" that represents a local aspect of the strategy. Initially, the agent does not give any advice. The refresh strength for the $P_F$ pheromone state="A" is zero. Without any advice, the control system simply operates in the basic behavior defined in the reactive layer. A 50/50 split of the flow at "S1" results and the load at "P1" stabilizes at 3.125 workpieces. These values are predicted and validated already in Section 4.5.3.

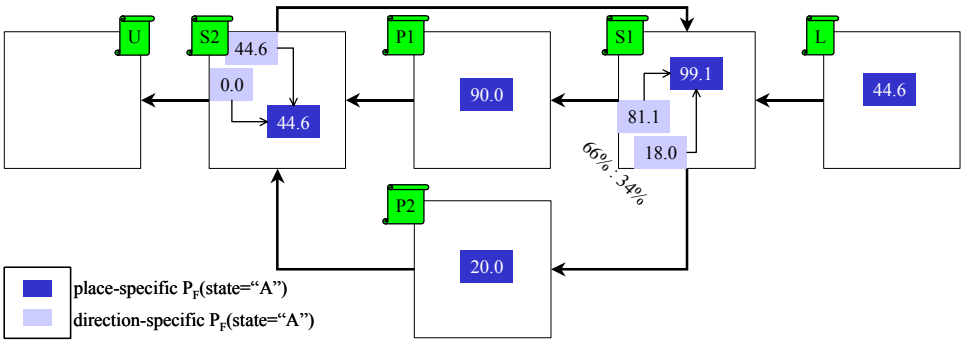


Figure 4.21. $P_F$ Pattern

The adaptation process of the Policy-agent perceives a load value that is too low. The agent needs to attract more workpieces. The advice is changed towards positive $P_F$ in the regular refresh. The change of the advice occurs very slowly and gradually, because it takes a while for the change in the advice to become visible in the $P_F$ pheromones, and some more time has to pass before the change in the material flow actually manifests itself in $P_S$ and $P_P$. The dynamics of the adaptive process may be tuned manually on the basis of the formal model. Alternatively, a further process may be added that adapts the dynamics of the adaptation process to the system at hand.

The adaptation process changes the advice so that the observed material flow eventually matches the current strategy. In the demonstration, the Policy-agent at “P1” refreshes the $P_F$ pheromone state=“A” finally with a strength of seven every second. The resulting $P_F$ pattern is depicted in Figure 4.21. The $P_S$ strength at “P1” stabilizes at forty, encoding a load of four workpieces at “P1” as it was required by the strategy.

The goal to achieve and maintain a load of four at “P1” results in a 66/34 split of the flow at “S1”. More workpieces are processed at “P1” than at “P2”, improving the combined yield of the two processing units to a “A”:”B”=17:83 ratio, and reducing the flow back from “S2” into “S1”. The required overall processing capacity is reduced to 1.2 workpieces per minute instead of 1.25 as it was without the advice.

# Chapter 5 - Future Research I

*The GMC system presented in Chapter 4 accepts a strategy and tries to implement the required material flow pattern. The following chapter proposes detailed concepts for the automatic evaluation of the performance of a strategy in terms of global production goals and concepts for the automatic generation of new strategies (Section 5.1). Furthermore, an integrated approach to criticality-driven visualization is presented (Section 5.2).*

*The purpose of Chapter 5 is to provide a starting point for future research. Even though there are agents and pheromones proposed in the following sections already, there has to be more research invested to realize and validate the concepts.*

## 5.1 Automatic On-Line Optimization

System operators run complex production systems like a paint-shop based on flow-patterns. The basic tasks are executed by local control (manual and automatic) at the shop floor. The goal of the operators is to provide the global perspective and guide and coordinate the local task execution. Their advice is given in terms of local patterns (e.g., "At the switch XYZ, $x_1$ percent of all "spot-repair" cars go to exit one and the rest goes to exit two"). The local control has to translate the advice into control decisions for the single car.

The *GMC* system enables the operators to run the production system as they did before. The system takes care of the automatic translation of the advice into control decisions according to the current situation in the plant. The control system makes effective use of the flexibility and robustness of a production system and, at the same time, it provides the user with an intuitive way of interfering with the operation.

An automatic on-line optimization adds more layers to the advisory system, supporting an operator by selecting and even generating strategies. Instead of just executing a given strategy, the extended system tries to evaluate the effectiveness of the strategy in achieving production goals. Learning from past experience, the system accumulates strategies and their situation-specific evaluation. The space of possible strategies is searched for better performing ones at the highest layer of the advisory system.

An approach to the evaluation of the currently implemented strategy in terms of production goals is presented first. The strategy evaluation layer is the second layer of the advisory system placed on top of the strategy implementation layer. A third layer is introduced to realize strategy ranking and generation.

### 5.1.1 Strategy Evaluation Layer

With the introduction of the strategy evaluation layer multiple strategies are handled by the advisory system. One of these strategies is the currently implemented one while all the other strategies are passive strategies. In the following, the currently implemented strategy is called the current one for short.

The activities of the Policy-agents in their attempt to implement their goals change the

execution of the production. The consequences may be positive or negative in terms of the usually globally defined production goals. The requirements specified by these goals have to be met by the *GMC* system as a whole and not by a single agent. Since the behavior of the system emerges from the actions of its agents, the evaluation of the goal fulfillment must emerge too.

A high global throughput of the production system is the most important goal in the paint shop. Global throughput is measured in the number of workpieces leaving the production system successfully processed in a fixed period. Therefore, it is sufficient to place simple agents at the exits of the system and have them count all passing Workpiece-agents. Other production goals require other methods of measurement.

When the current goal fulfillment is accessible to the agent system, the credit assignment problem is still left to be solved. The perceived output is a result of the current material flow. But, does the observed pattern match the goal of the current strategy? In other words, is the currently implemented strategy to be given credit for the current production goal fulfillment? Implementing its own local aspect of the strategy, each Policy-agent perceives the local difference between the current material flow pattern and its goal pattern. According to the perceived difference, the agent changes its advice. Thus, a Policy-agent not changing its advice must have met its goal.

The proposed emergent evaluation of a strategy in terms of production goals is related to resource-based approaches in multi-agent coordination. Each strategy in the advisory system is assigned an account. The account of the current strategy is filled continuously according to the fulfillment of production goals, and it is reduced by Policy-agents adapting their advice.

In terms of production goals, strategies are successful if their account rises and they are unsuccessful when it falls. The degree of change in the account of a strategy provides a rough estimate of the quantitative evaluation of the strategy. But, it is not more than an estimate since the proposed resource-based approach operates on a very high level of abstraction. Figure 5.1 illustrates the relations among the elements of the resource-based evaluation of a strategy.

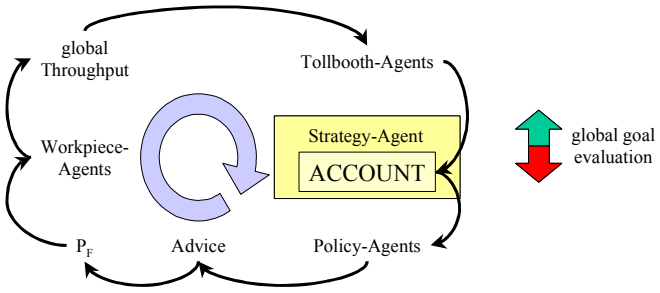


Figure 5.1. Feedback Loop in the Resource-Based Strategy Evaluation

The implementation of the resource-based approach to the evaluation of a strategy in the example of global the throughput production goal requires two additional agent types. There are Tollbooth-agents, translating the local goal fulfillment into an input to the account of the current strategy, and there are Strategy-agents, representing a strategy and managing its account. Furthermore, the specification of a Policy-agent is extended to transmit its resource-usage when it changes its advice.

#### The Tollbooth-Agent Type

A Tollbooth-agent provides the resource input to the current strategy. A resource-input is directly linked to the fulfillment of the goal of high global throughput. The agent type derives its name from the fact that Workpiece-agents meeting a Tollbooth-agent have to "pay a toll" if they represent a finished workpiece. A Tollbooth-agent is usually co-located with an Unloader-agent.

After it is set up, a Tollbooth-agent requests from its Place-agent a notification when a Workpiece-agent arrives at the place (Section 3.3.2). Upon receiving such a notification, the Tollbooth-agent contacts the Workpiece-agent, requesting the processing state of the workpiece. If the state matches one of the specified final states, the Tollbooth-agent provides an input to the account of the agent of the current strategy by sending a message.

The current Strategy-agent is known to the Tollbooth-agents by reference. The restriction to local communication only is broken to keep the design simple. The interaction does not take up much bandwidth in global communication.

#### The Strategy-Agent Type

A Strategy-agent represents one strategy. If the strategy is the current one, its implementation is attempted by its Policy-agents. Each strategy has its own set of Policy-agents, which are known to the Strategy-agent by reference.

The account of the strategy is handled by its Strategy-agent. The agent receives positive inputs to the account from Tollbooth-agents. The sequence of inputs represents the current level of the throughput-goal fulfillment. The account is reduced with every change of advice by one of the Policy-agents. Every time a Policy-agent changes its advice, the strength of the change is used to compute the resource usage. In a message, the Policy-agent tells its Strategy-agent by how much the account is to be reduced.

#### Extended Agent Behavior

To cooperate within the strategy evaluation layer, Policy-agents have to extend their cyclic adaptation of their advice. After the required changes are computed, the resulting resource usage is determined. The resource usage is proportional to the absolute strength of the change. A Policy-agent sends the amount of the resource usage to its Strategy-agent.

Also the behavior of the Workpiece-agents is extended, permitting the Tollbooth-agent to request the current processing state of the workpiece.

### 5.1.2 Strategy Ranking and Generation Layer

The strategy ranking and generation layer supports the human operator in the selection of the currently appropriate strategy. Its first task is to provide a ranking of a given set of strategies according to the current situation in the production process. The second task is to explore the space of possible strategies and to change the currently available ones.

**Strategy-Ranking-Net**

The agents realizing the adaptive ranking of a set of strategies are organized in a spreading activation network called Strategy-Ranking-Net. The abstract model of the net comprises three types of nodes (Figure 5.2). There are nodes matching the places in the *PI*, there are nodes representing the elements in a classification of the pattern segments at each place, and finally there are nodes for each strategy.

A classification of the locally perceivable segment of a material flow pattern is required, because potentially, there may be an infinite number of patterns. A (small) set of pattern classes permits reasoning on a range of patterns. A local classification should be designed to assign the most often encountered patterns in the actual production system to different classes.

The nodes of the places are linked to the nodes of the local pattern classes. The links are not weighted. Each pattern class node is linked to each strategy. These links are labeled. The label, a real number between zero and one, represents the current applicability of a strategy considered for a specific place and pattern class.

In the network model, a ranking of a set of strategies specific to one flow pattern is realized by a spreading activation starting at the nodes of the places. For each place the respective node of a pattern class that is assigned to the considered flow pattern is activated. From there, the activation spreads via the labeled links to all strategy nodes. This last step sees a reduction of the propagated strength according to the applicability a strategy to the flow pattern. The label of the link represents the applicability.

The activation arriving at a strategy node adds up into an internal activation value. After the propagation process is completed, the ranking of the strategies is given through their activation values. The largest activation value indicates the highest rated strategy for the considered flow pattern.

The network model is adapted over time by changing the relevance labels according to the performance of a strategy in the face of the current flow pattern. The performance evaluation considers production goals.

In Figure 5.2 the abstract model of the Strategy-Ranking-Net is shown. Shaded in the background the mapping of the entities of the model to agents is indicated.

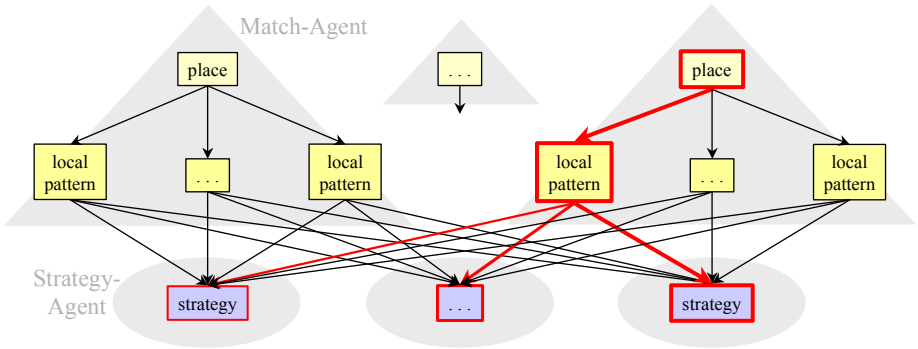


Figure 5.2. Strategy-Ranking-Net Model

The model of the Strategy-Ranking-Net is closely related to the Case-Retrieval-Net model, which is applied in case-based reasoning for efficient organization of a case base. The Case-Retrieval-Net has been developed at Humboldt-University Berlin [Lenz and

Burkhard, 1996a] [Lenz and Burkhard, 1996b] and it is applied to a number of practical problems (e.g., technical diagnosis in [Lenz et al., 1996]).

The information-entity nodes (attribute-value pairs describing one aspect of a case) of a Case-Retrieval-Net relate to a pair of one place node and one pattern-class node, whereas the case nodes equal the nodes of strategies. With all "case-descriptions" being structurally the same (each place-pattern pair is linked to each strategy), the "cases" are only distinguished in the specific relevance of a strategy for a pattern. The basic retrieval process in a Case-Retrieval-Net operates in the same way as in a Strategy-Ranking-Net.

The adaptive Strategy-Ranking-Net is realized by the interaction of three agent types. There are Match-agents representing the place-pattern pairs for one place in the infrastructure. Then, there are Strategy-agents fulfilling the role of a strategy node in the model. Finally, there are Policy-agents providing evaluation information for the adaptation of the ranking.

The relevance labels are adapted continuously according to the current performance of a strategy. Furthermore, there may be requests for a ranking of the currently available strategies coming from a user at any time.

A Match-agent is able to perceive and to classify the local segment of the current flow pattern at its place. Furthermore, the agent manages the currently assumed relevance of each available strategy for each local pattern class. The Match-agent knows the strategies by reference to the Strategy-agents.

When a ranking of the currently available strategies is requested from a Match-agent, it accesses the current flow pattern and maps it to one of the pattern classes. The pattern class provides the agent with the current relevance of each strategy. Each Strategy-agent is sent an activation value computed by the multiplication of the relevance value and the activation specified in the request. Finally, the sender of the request is notified that the task of the Match-agent is fulfilled.

The Strategy-agents add up the received activation values in an internal variable. When their activation is eventually requested, they pass it on and reset the variable to zero.

The following steps have to be taken to retrieve the ranking of strategies for the current flow pattern. After making sure each Strategy-agent carries zero activation, all Match-agents are requested to spread activation according to their adapted evaluation. When confirmations have been returned from all Match-agents all Strategy-agents are asked for their activation. The strength of the activation determines the ranking of the strategies.

To realize the adaptation of the relevance data held by the Match-agents, the gap between the changes of the account handled by the Strategy-agents and the Match-agents has to be bridged. The reinforcement learning approach requires the Match-agents to know the performance evaluation of the currently implemented strategy. The evaluation is generated resource-based and it presents itself in the change of the account of the strategy. An increasing account indicates a good strategy; a decreasing account is attributed to a bad strategy.

The performance of the current strategy is transmitted to the Match-agents using a pheromone type "Evaluation" ($P_E$). Inputs to $P_E$ do not propagate. $P_E$ carries one additional data slot referencing a Strategy-agent. Policy-agents implementing a strategy perceive the change of the account of their strategy over time. The $P_E$ pheromone of the current strategy is refreshed proportional to the strength of the change.

The behavior of the Policy-agents is extended by one more cyclic process. A Policy-agent regularly accesses the current status of the account. Comparing the current status to the one retrieved in the previous cycle, the Policy-agent determines the refresh strength to be used from the difference of these two values. After the new account status has been stored for the use in the next cycle, the Policy-agent refreshes the $P_E$ pheromone matching its strategy.

Using a pheromone-based transmission of the evaluation, Policy-agents are decoupled from Match-agents. The separation is necessary because the agents operate in different time-scales. Furthermore, the computational power of the agent environment is tapped since the strength of the $P_E$ pheromone matching the current strategy approximates the change of the account over time (first deviation).

A Match-agent runs an additional cyclic process to realize the adaptation of the relevance data. In regular intervals, the strength of the $P_E$ pheromone of the current strategy is sampled. According to the perceived strength, the relevance label of the link from the current pattern class to the strategy is changed by a very small value. Positive pheromone strength reinforces the links. Negative strength weakens future propagation of activation.

### Evolutionary Strategy Generation

Whereas the spreading activation network of Match-agents and Strategy-agents ranks the current set of strategies, a further extension generates new strategies, realizing an ongoing exploration of the infinite search-space of strategies.

An evolutionary search for strategies takes the fitness of the currently available strategies into account. New strategies are generated from the recombination of fit strategies including mutational changes. Strategies are deleted from the system if their fitness is low.

The fitness of a strategy is based on the relevance to the current flow pattern in the production system as perceived by Match-agents. But while Match-agents only perceive the local relevance of a strategy, the Incubator-agent considers its global fitness. The Incubator-agent is the central element of the evolutionary strategy generation. It is not located at any of the places in the *PI*. Instead it communicates directly with all local Match-agents. At regular intervals the Incubator-agent requests the complete mapping of pattern classes to strategy rankings from each Match-agent. The returned data is aggregated into one numerical fitness value for each currently available strategy.

In a probabilistic selection, the Incubator-agent chooses strategies for recombination and strategies for extinction. Strategies with a higher than average fitness have a higher probability of being selected for recombination. Weaker strategies are more often selected for extinction. The recombination of strategies into a new one may operate on different levels of detail. The new strategy could be a selection of pattern segments from the parent strategies (high level recombination). Or the local load requirements may be recombined (low level recombination). Mutation only changes load values.

In one Incubator-agent cycle there is always the same number of extinct strategies as there are new ones generated. The Incubator-agent executes a genetic algorithm with a constant population size. The representation of the genetic code is not a bit-vector, but a hierarchy of places and patterns.

To ensure a stable operation of the advisory system, the Incubator-agent runs at a much slower rate then the other agents. Strategies need time to be implemented and evaluated in different scenarios before they have a meaningful fitness value. Furthermore, there should be a basic set of strategies defined by the operator, which the Incubator-agent is not permitted to select for extinction.

## 5.2 Visualization

Visualization is another important issue in the extension of the *GMC* system. With the distribution of the actual control over the production system, there is no central point of access of data on the current operation. Furthermore, because of the complexity of the ongoing operation human operators are not able to understand and influence the processes when provided only with raw data. Aggregation and filtering is required.

In the following, a third sub-system is proposed. The visualization system operates on information provided by the control system, by the interface layer, and by the advisory system. The visualization system provides an interface to the human user. Its agents generate specific local and global views on the current state of the productions system, providing the user with up-to-date information, aggregated and presented in a human-friendly fashion.

Presenting the human operator with up-to-date information focused on current points of interest in location and aggregation is a challenging task, most of today's state-of-practice systems are not yet fully up to. The pheromone-enhanced synthetic ecosystems approach presents the designer of the system with an opportunity to integrate high-quality visualization seamlessly.

The different components of the *GMC* system generate a lot of locally available data on the current operation of the production process. The available information concerns different levels of aggregation (workpieces, flows, strategies), and different time scales (control system versus advisory system). There is data representing the past, present, and an approximation of the future of the production process. Some of the available data is directly accessible in specific pheromones (e.g. processing capabilities). Other information has to be requested from the agents (e.g. strategy evaluation).

The visualization system operates in two modes. In the “surf” mode, the user specifies the current point of interest (focus) and the required abstraction. In analogy to the Internet, the operator “surfs” the information in the production system. The user may point the visualization to different “sites” (places) and access specific “pages” (local information / local view) there.

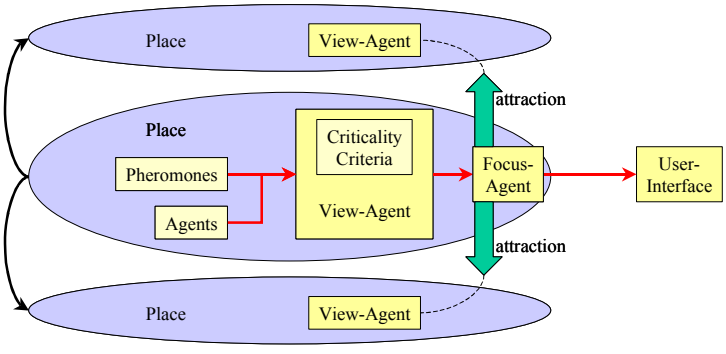


Figure 5.3. Pheromone-Based Focusing in the Visualization System

The second mode of visualization is the “auto-pilot” mode. The visualization system automatically relocates the focus of the user towards critical situations when triggered by events inside the other layers. The user initially gives the criteria for criticality. An adaptive visualization system re-evaluates its criteria constantly, analyzing the reaction of the user to the presented view.

Figure 5.3 illustrates the proposed realization of the visualization system. One possible focus of the view of a user into the production system is represented by a Focus-agent. Focus-agents constantly move through the *PI*. The pheromone type “View” ($P_V$) transmits attracting forces that guide the movements of the Focus-agents. Whenever a Focus-agent finds a point of interest, it triggers the user interface to focus on the current place of the agent.

The generation of the local view on the production process is the task of the View-agents. For each place in the *PI* there may be a View-agent. A View-agent accesses the available information and it aggregates this information into a set of status documents. When a Focus-agent migrates to a place of a View-agent and if the local criticality level is sufficiently high, then the Focus-agent transmits the documents of the View-agent to the user interface.

It is the task of the View-agents to gain the attention of the user in the “auto-pilot” visualization mode. View-agents generate attracting forces influencing the movement of the Focus-agents. The generation of the forces is based on the criteria of criticality of the respective View-agent. The more critical a local situation appears to a View-agent, the stronger is the “interest” of the agent to attract Focus-agents. In very urgent situations the View-agent spawns a new Focus-agent if it is not able to attract one of the existing ones in time.

There are several advantages of approaching the visualization of the system in such a distributed manner. The operation remains decoupled from the monitoring, communication of the process status is reduced to points of interest, and data is aggregated according to the requirements of the user.

In comparison to state-of-practice visualization systems, the main advantage of the approach is the guidance of the focus of the user on the basis of critical situation previously defined by the user or even learned by the system. The user is always provided with a ranking of the criticality of different foci. Based on its own experience the user selects a view from the presented ranking. Thus, the system may re-evaluate its ranking parameters and learn the preferred view on a place.

In addition to the display of the operation, the visualization system may also present the user with suggestions for strategies to be implemented according to the results of the strategy ranking and generation layer. In this case, the user may enter new strategies and change existing ones. Or, the user may change their relevance for specific flow patterns. Finally, the user may require the system to begin implementing a different strategy. Thus, the user operates on a high-level abstraction of the production process.

# Chapter 6 - Synthesis

*Chapter 3 proposes and generally motivates an extensive set of design principles for synthetic ecosystems. Furthermore, it suggests, formalizes, and analyzes the pheromone infrastructure, an extension of runtime environments of software-agent systems that supports sign-based stigmergy. The Chapters 4 and 5 present the guided manufacturing control system, whose design follows the proposed principles and that makes use of the PI in its multi-agent coordination mechanisms.*

*The following chapter returns to all three major subjects. First, for each design principle the resulting characteristics of the GMC system are discussed systematically (Section 6.1). Then, the importance of the PI for the evaluation of emerging system-level behavior and modes of information sharing through the active agent environment are considered (Section 6.2). Finally, the operation of manufacturing control systems is discussed in abstract state space and conclusions for the GMC systems are drawn (Section 6.3).*

## 6.1 Re-Visiting the Design Principles

The following discussion of the design principles further motivates their respective relevance for the design of large-scale self-organizing systems. The consequences and advantages for the *GMC* system and the applicability to manufacturing control in general are considered for each principle.

### 6.1.1 Single-Agent Principles

**Things, not Functions (SE-Principle 1)**.—The reactive layer is the only part of the multi-agent system that has direct control of the physical environment. All other layers act only through the reactive layer. Thus, the advice to model agents according to real-world entities is only applicable to the design of the reactive layer. The other layers introduce agents on the basis of concepts like a strategy, a policy, or a user-focus. But it is never a function that is identified with an agent.

All agents in the reactive layer represent real-world entities. The resources in the manufacturing system are controlled by Loader-agents, Switch-agents, Processing-agents, and Unloader-agents. Workpieces are represented by Workpiece-agents. The composition of the agent system mirrors the current configuration of the manufacturing system.

One major advantage of the chosen way of modeling the reactive layer is the resulting simplicity of the change management. Whenever a resource in the manufacturing system is added or removed, the corresponding change is made in the agent system. Following the strict locality of the direct agent interactions (SE-Principle 9), such a change does not have to be told anyone else but the agents in the close neighborhood. Additionally, the automated information sharing mechanisms make sure that whenever information about resources are needed farther away, the change is automatically published.

What would have been the design of the reactive layer if the agents were modeled according to functions? Functions in the reactive layer are for instance transportation or

resource management. Assume there is an agent for transportation management. Such an agent would have to control the transport of every workpiece in the system simultaneously. While such an approach may be feasible in very small and simple transport systems, scaling up the system, for instance to the size of a real paint-shop, would require too much communication and too many decisions at the same time. No software system would be able to handle the load and hence, the transportation management agent would become a bottleneck in communication and computation.

**Small Agents (SE-Principle 2)**.—Most agents in the *GMC* system are small in comparison to the whole system. A Ghost-agent is an example for small agents. The lifespan and the actual impact of the actions of one Ghost-agent are neglectable. They do not control anything and a single Ghost-agent by itself does not make anyone else do something. But the advisory system depends in its operation on the information they jointly generate.

In general, the agents are very simple to implement. Once an agent type is completely specified, it is just a matter of hours to implement it and test it for correctness according to the specification. Considering again the Ghost-agent as an example, agents of this type are programmed to handle two interactions consisting of one “ask” and one “reply” message altogether. Ghost-agents run just one process. They recognize two different pheromone types and they refresh a third type themselves. The decision process of a Ghost-agent is straightforward and easy to understand, even though it is one of the most complex processes in the system already. Other agent types run more than one process in parallel, but still they are all very simple software artifacts.

With the simplicity of implementation of the agent system, much more effort may be invested into design, tuning, and the thorough testing of the behavior of the whole system.

**Diversity, Heterogeneity (SE-Principle 3)**.—A simple and intuitive way to design small agents is to diversify the agents in accordance to specific responsibilities. Each agent gets allotted only a small set of responsibilities. Agents of the same type share the same general responsibilities.

In the *GMC* system, a general division of responsibilities identifies the different sub-systems and layers. The control system, the interface layer, the advisory system, and the visualization system all fulfill different functions. The agents inside a layer are given specific responsibilities. For instance, it is the responsibility of a Tollbooth-agent to report all passing workpieces that have been completely and successfully processed. Tollbooth-agents are extremely simple, because they only report these workpieces but do not process the information any further.

Diversified according to their specific responsibilities, the agent types all require specific programming. But, as it is already shown in the example of the Ghost-agent, a small number of actual responsibilities lead to a simple implementation. Furthermore, maintenance of the control system requires less effort. Whenever elementary responsibilities change, the components that must be changed are small and easy to identify.

To reduce the complexity of the presentation of the *GMC* system, heterogeneous agent types have not been specified. But, there is still potential for heterogeneity. For instance, the Workpiece-agent type may be split into different families when normal orders and rush orders are handled separately. Then, all Workpiece-agents still engage in the same interactions and the exchanged messages are still the same, but the internal decision processes of the agents run differently.

Besides parallel heterogeneity represented in different families of the same agent type, sequential heterogeneity in generations of agents of the same type is another option in further versions of the *GMC* system. Consider again the Workpiece-agent. If the system operates for a long time in the same manufacturing system, Workpiece-agents could adapt their decision processes to the specific layout of the transport system or to the characteristics of the processing resources. The adaptation affects all Workpiece-agents in parallel, spawning new generations.

In general, agent types most prone to heterogeneity are those that have families with a distinctively different bias in their decision processes (e.g. normal order vs. rush order), or agents that have a comparatively short life cycle to change over generations.

**Redundancy (SE-Principle 4)**.—The only component in the *GMC* system that includes redundant elements is the visualization layer. There are multiple Focus-agents, which wander through the system. They look for View-agents that consider their current documents sufficiently important to present them to the operator. The paths of the Focus-agents are not specified beforehand. They move randomly, guided by attracting pheromones sent out by the View-agents. Thus, redundancy makes sure that on average each View-agent is visited often enough to guarantee (statistically) a required response time of the overall visualization system.

But, redundancy may be incorporated into many other parts. For instance, each Workpiece-agent should be able to handle more than one workpiece at the same time to increase the stability of the system. Then, take-up mechanisms are included into the agents. They make sure that if a Workpiece-agent fails others pick up its tasks. Such redundancy is required when the system is actually fielded.

### 6.1.2 System-Architecture Principles

**Decentralization (SE-Principle 5)**.—Most components in the *GMC* system are designed following the decentralization principle. Elements of centralization are only found in the higher layers where the manufacturing system is considered a single unit. There are for instance the Strategy-agents. Exactly one of them is active at a time, coordinating the evaluation of its strategy in its distributed implementation by the Policy-agents.

In the lower layers of the system, all agents operate decentralized. There is no global White Pages service required and local services are realized by the Place-agents of the *PI*. Teams are formed dynamically whenever tasks arise. The members of a team join in on their own accord.

**Modularity (SE-Principle 6)**.—The *GMC* system is grown layer by layer. The layers and their respective sub-systems are identified according to major functionalities of the system.

The reactive layer is the core of the system. All by itself it realizes the completion of the incoming orders, making sure transport and production constraints are fulfilled. These constraints are set by the processing requirements of the products, by the layout, and by the capabilities of the manufacturing system. As specified in the SE-principle, the core part of the system is able to operate stand-alone.

The other layers add functionality to the global system behavior without compromising the correctness of the operation in the manufacturing environment. None but the reactive

layer is in direct control of the manufacturing system. All the agents of the other layers can do is advising the reactive layer in its operation. It is up to the agents of the reactive layer to heed or to ignore the advice. It is their responsibility to maintain a correct system behavior.

The advantage of ordering the components of the system into separate layers instead of arbitrary parts is to be found in the clear interdependencies of the layers. In general, each layer requires the availability of all layers below it. Thus, the start-up sequence of the system is clearly defined.

Modularization of complex software systems is not specific to the synthetic ecosystems approach. Modular systems design has been successfully used for many years now. Component-based approaches promote fast and efficient realization of large software systems through the combination of customized standard components, promising extended re-use and reduction of development costs. Thus, following the modularity principle integrates a successful general principle in software design into the synthetic ecosystems approach.

**Parallelism (SE-Principle 7)**.—Parallelism is introduced by the manufacturing system itself. Manufacturing is inherently parallel. Transport and processing tasks arise throughout the system in an asynchronous manner. Thus, a system controlling the production has to match this parallelism.

Teams are formed dynamically in the reactive layer. There is always a Workpiece-agent and a Resource-agent teaming up to fulfill tasks that concern a workpiece. As long as a Workpiece-agent manages only one workpiece (see redundancy), the agent is engaged in one interaction at a time only. The Resource-agents, on the other hand, are designed to manipulate multiple workpieces. They join more than one team at a time and they interleave separate interactions with different Workpiece-agents.

At the same time Policy-agents pick up the pattern information generated by the Workpiece-agents and Ghost-agents and give their advice according to the local aspect of their global strategy.

**Bottom Up Control (SE-Principle 8)**.—With a multitude of small agents interacting to control the underlying manufacturing system, the bottom-up-control principle is consequently followed in the *GMC* system. There is no single entity controlling the whole transport system or the whole manufacturing process by itself. No agent has global access or even the ability or knowledge to run the system.

Teams fulfilling small tasks are dynamically formed on the basis of the local expertise of the agents, their responsibilities, and their goals. As a consequence of these many activities, material flow patterns emerge. Agents at higher layers of the control system observe these patterns and base their decisions on them. The higher layers try to influence the lower layers. But, the actual control lies at the bottom – at the core of the *GMC* system.

### 6.1.3 Interaction Principles

**Locality (SE-Principle 9)**.—Locality is strictly followed by all agents operating in the *PI*, which provides a spatial structuring of the agent system. The places in the *PI* map to elementary zones of the manufacturing system. A Resource-agent controls all elements in its zone. Agents that represent mobile entities (e.g., Workpiece-agent, Ghost-agent) move from place to place. But, wherever they are, they are only permitted access to pheromones

at their current location, and they only interact with agents sharing the current place. According to the current situation and local information the agents decide dynamically which direction to go to and what agents to interact with as they explore the system.

There is no agent that has global knowledge about the system. Even singular agents outside of the *PI* like Strategy-agents or the Incubator-agent know only specific aspects of the overall system.

**Indirect Communication (SE-Principle 10)**.—Indirect communication among agents in the *GMC* system is used, wherever:

- information generated by another agent at a different location is required for a decision of an agent, e.g.,
  - ability information of Processing-agents and Unloader-agents attracting the Workpiece-agents towards their locations
  - criticality information of View-agents attracting Focus-agents
- a decision process of an agent requires information that is jointly generated by other agents at the same location, e.g.,
  - current load at the place generated by all Workpiece-agents present
- the sender and the receiver of data operate on different time-scales, e.g.,
  - predicted load at the place generated by the Ghost-agents and processed by the Policy-agents
  - performance evaluation generated by the Policy-agents and processed by the Match-agents.

The *PI* is the channel for the indirect communication of data. For each transmission a pheromone type is specified. The provider of information refreshes selected pheromones in a specific way and thus the receiver may deduce the information from the sampled pheromone strength.

The design, evaluation, and the tuning of indirect communication mechanisms is supported by the results gained in the analysis of the formal model of the pheromone infrastructure (Section 3.2). For any given topology and any refresh behavior, the resulting local pheromone strength may be approximated. The more regular the refresh behavior is, the more accurate is the approximation.

**Recursion, Self-Similarity (SE-Principle 11)**.—The regular refresh of pheromones is a generic principle used throughout the whole system to communicate and aggregate data in the *PI*. Regular refresh behavior is preferred since it allows a very accurate analytic prediction of the resulting pheromone patterns, reducing the error in the interpretation of the sampled data.

The use of pheromones for spatial coordination of movements of agents is a second mechanism found in different layers of the system. Examples are the attraction of Workpiece-agents and Ghost-agents by Processing-agents or Unloader-agents, or the attraction of Focus-agents by View-agents.

Structural recursion or recursion in mechanisms is not realized in the *GMC* system, because there is no singular agent at one level that represents an aggregation of multiple agents at another level. The only structure coming close is the representation of a strategy during implementation by the Policy-agents and during evaluation by the Strategy-agent. But then, completely different interactions occur at the different levels. More depth in

structure and more similarities among the different levels are required to make effective use of recursion.

**Feedback, Reinforcement (SE-Principle 12)**.—Reinforcement and feedback play important roles in most coordination mechanisms of the *GMC* system. Reinforcement is used, for instance, to bias the system towards successful strategies in terms of goal fulfillment and to steer the system away from unsuccessful strategies (Section 5.1.2). The performance of the currently implemented strategy reinforces the bias towards or away from a strategy.

Feedback effects are included when the current strategy is implemented by its Policy-agents. Ghost-agents, regularly created by Workpiece-agents, generate a prediction of the local load pattern as they move downstream and approximate a possible future behavior of their Workpiece-agent. Each Policy-agent adapts its advice according to the local prediction. The advice is transmitted in pheromones that are propagated upstream. The path of a Ghost-agent is influenced by the pheromones it encounteres. Changed advice changes the predicted pattern, until the predicted pattern matches the goal pattern of the Policy-agent. There exists a closed loop of cause and effect, tuning the advice. The Workpiece-agents follow the advice in the same way their Ghost-agents do, making the prediction come true.

The adaptation of behavior on the basis of a repeated re-evaluation of a prediction is found in many coordination and control mechanisms in nature. Humans do it too, for instance when preparing to catch a ball.

**Randomization (SE-Principle 13)**.—The application of feedback effects in synthetic ecosystems becomes very hard when no randomization in the internal decision processes of the agents is introduced at the same time. Consider again the adaptation of the advice of the Policy-agents, but now in a scenario with two Policy-agents at different places ($x$ and $y$) acting on the same upstream place ($z$). The two Policy-agents both want some share of the flow of workpieces out of place $z$, but they both do not want the full flow.

Consider the decision process of Workpiece-agents and Ghost-agents at place $z$ when choosing between the direction towards place $x$ and towards place $y$ but assume that the selection is not randomized and they always choose the direction from where the strongest attraction comes. For the sake of simplicity assume furthermore that the agents all go towards place $x$ if the difference between the two competing forces is zero.

There is no simple and continuous solution to the presented problem. Either, the Workpiece-agents all go to place $x$, or they all go to $y$. Only when the Policy-agents change their advice very fast and synchronized, they may split the flow in the required manner. But then, they would have to operate on the same time-scale as the agents of the reactive layer do, since they are then primarily concerned with the single workpiece instead of a flow of workpieces.

The discussion already shows how randomization prevents thrashing effects, which make an intuitive, stable, and continuous control of the system hard or even impossible to achieve. Other examples are found in the routing decision of Workpiece-agents selecting processing resources in the reactive layer, the generation of the predicted pattern in the interface layer, or the evaluation of strategies. It remains an issue for future research, to assess if the randomization of the internal decision processes of the agents increases the entropy on the micro-level, and thus stabilizes the global operation of the system (macro-level), as proposed metaphorically in [Parunak, 1997].

**Evolutionary Change (SE-Principle 14)**.—The adaptation of the advice of a Policy-agent is considered again. A Policy-agent perceives the difference between the current or the predicted local pattern and its goal pattern. The agent changes its advice by a small and fixed amount according to the perceived difference. Thus, this difference only determines the direction of the change but not its strength.

There are two primary motivations for the chosen design. First, the change in the advice takes some time for its effect on the patterns to become visible. And second, the perceived difference could be an abnormal perturbation.

The first reason, the time lag between the change and the effect, introduces a necessary delay into every complex adaptive process where the absolute effect of the change cannot be predicted beforehand. The Policy-agent can do nothing but wait to see how the changed advice influences the flow, especially, if there are other Policy-agents nearby that also influence the flow and that also change their advice continuously. In this case, abrupt changes lead only to thrashing and any sensible guidance is lost.

The nervousness of the agents is reduced when perturbations in the data is ignored. All data they perceive is noisy. Only lasting change in the material flow should be adapted to. Again, the Policy-agents should not change their advice with every workpiece that is running late. The Policy-agents operate on a larger time-scale. Their speed of change must be much slower for the single workpiece to lose itself in the aggregated flow. The use of pheromones is a means for such long-term aggregation. While they slowly evaporate, dozens or even hundreds of Workpiece-agents have left their tiny marks on the trail.

**Information Sharing (SE-Principle 15)**.—Information sharing is encountered throughout the components of the *GMC* system and much more could be integrated yet. The use of pheromones to coordinate many small agents demonstrates system learning. Adaptation of the advice of a Policy-agent is an example for individual learning. Sharing information between two generations of agents is mentioned in the discussion of SE-principle 3 where adaptation of decision parameters of Workpiece-agents to the given layout of the manufacturing system is proposed.

System learning dominates especially in the design of emergent features in synthetic ecosystems made up of a multitude of simple agents. But, some form of global coordination is required to design emergent features. A very powerful tool for global coordination in synthetic ecosystems is the use of pheromones as shown in the *GMC* system.

**Forgetting (SE-Principle 16)**.—Learning and adaptation in long-lived systems that act in a continuously changing environment improves performance only when combined with some form of forgetting. As long as no information is ever given up, the system and its components eventually suffocate under the load of accumulated data. It is necessary to lose old knowledge to make space for new.

The evaporation of pheromones in the pheromone infrastructure of the *GMC* system is the most obvious forgetting mechanism in information sharing. But it is only effective because most information used in system learning is contained in pheromones and the pheromones are used to bias otherwise random choices.

The eventual death of each adaptive agent is the second mechanism providing the system with forgetting. Agents permitted to live “forever” are the Resource-agents, which do not adapt. Most other agents potentially die. The agents with the shortest life expectancy are the Ghost-agents, followed by the Workpiece-agents. Policy-agents die

when their strategy is no longer implemented, and Strategy-agents die "by the hand" of the operator or they are killed by the Incubator-agent if they do not perform. With the potential death of each Strategy-agent, all data contained in the Match-agents is eventually replaced. "Immortal" agents like the View-agents, Focus-agents, and the Incubator-agent have no adaptive mechanism attached to them.

**Multiple Goals (SE-Principle 17)**.—The *GMC* system is designed to suffice maintenance-goals. In the presented form there are two major goals:

1. completely process every incoming workpiece according to its processing needs
2. change the material flow towards improved global throughput

Other maintenance-goals may be integrated into the behavior of the system at any time. The suggested evaluation of the goal-fulfillment is open and intuitive. But, it is impossible for the system to fulfill a global achievement goal, primarily because the life of the system has no specified end-point where the fulfillment of the achievement-goal could be evaluated.

## 6.2 The PI – Agent System and Environment

Throughout the presentation in the Chapters 3 to 5, the pheromone infrastructure is considered as an extension of the runtime environment for software agents. In the following, the primary perspective is reversed and implications of an interpretation of the *PI* as an agent system are discussed.

Eventually, the agent system of the application returns into focus and the *PI* is "only" a part of the environment. Modes of information sharing among the agents are considered, presenting a broader and more abstract perspective than the previous chapters.

### 6.2.1 Analyzing Emergent Features

The *PI* is part of the runtime environment of agent applications and it provides the following generic and application-independent services (Section 3.3.2): "Access and Topology", "Local White Pages", "Pheromones", and "Notifications". The distributed implementation of the *PI* specifies the Place-agent type.

The community of Place-agents, which emulates an active pheromone environment for software agents, may be interpreted as a simple synthetic ecosystem: The Place-agents are small and simple; they only interact locally and share information. The information they share is encoded in pheromones, which evaporate and thus provide forgetting. Pheromones and pheromone patterns get their semantics in the context of the respective application, but the *PI* interprets a pheromone as what it originally is: a quantity of a chemical substance that is put into the environment to propagate and to evaporate.

There exists only a distributed specification of the system-level behavior of the *PI* that defines the behavior of the Place-agents and how they are linked. An evaluation of the system-level behavior requires an analysis of global features as they emerge from local interactions. For the time of the analysis of the *PI* (Section 3.2) the application agents (e.g., the *GMC* system) are considered the environment of the *PI*. Their behavior is only specified in terms of input patterns they generate in the *PI*. Furthermore, it is necessary to formally describe the internal Place-agent processes, the interactions among the agents of

the *PI*, and the influence of the outside environment (the application) to set up a formal model.

The *PI* is required to show global stability, which is a system-level feature. The formal model of the *PI* facilitates the proof of global stability. The model also permits the numerical prediction of emerging pheromone patterns in a wide range of input scenarios.

The ingredients required in the analysis of the features of the *PI* are also found in the synthetic ecosystem of an application that operates in the *PI* environment. The evaluation of the system-level behavior of the *GMC* system components follows a similar approach. To show a specific global feature, it represents the decision processes of all participating agents and their interactions among each other and with the *PI* in a formal model.

A numerical description of emerging multi-agent coordination is primarily supported by the use of indirect communication through the *PI* and the specification of many agent-internal decision processes based on probabilities determined by current pheromone patterns. The numerical description is further reduced in complexity when the SE-Principle 2 ("Small Agents") is followed in the design. If many agents are required to make an impact, statistical or probabilistic abstractions may be used in the description.

The numerical description of emerging multi-agent coordination is demonstrated in the evaluation of the *GMC* system in Section 4.5. There is, for example, the emergence of $P_S$ and $P_P$ patterns in the operation of the interface layer. These patterns are predicted numerically and they are actually observed in the simulation.

### 6.2.2 Modes of Information Sharing

The *PI* supports sharing of information among agents in an active environment. The specific characteristics of the *PI* permit different agent behaviors in the sharing process that have consequences for the transmission of information. In general, all information put into the *PI* is lost over time to provide an automatic forgetting mechanism for the agent system. An information provider has to take the evaporation of pheromones into account. The provider refreshes the stored information regularly, because it does not know when the consumer accesses the data. As a consequence, the refresh mode, which results in a specific pheromone pattern, becomes a transmitter of information in its own right.

There are two basic modes of information sharing when using the *PI*. Either information is shared locally within the local context, or the information is shared globally, outside of the local context.

In local information sharing the provider and the consumer are located at the same place in the *PI*. The provider regularly refreshes the information encoded in pheromones while the consumer observes the resulting pheromone patterns. In local information sharing the input events to the pheromones are not intended to propagate. The analysis of the formal model of the *PI* provides the designer with numerical predictions of the resulting patterns and facilitates tuning and evaluation. The transmission of the change of the account of a strategy sent by a Policy-agent to its co-located Match-agent is an example of the local information-sharing mode (Section 5.1.2).

There are two approaches when sharing information globally: active or passive information sharing. If information is shared actively, agents move through the *PI* and refresh pheromones according to the local context. In passive sharing the provider remains stationary but its input events are propagated to neighboring locations.

Passive information sharing makes use of the propagation mechanisms provided by the *PI*. The information provider does not have to care about how the information is spread. As a consequence, the designer may keep the agent very simple, because it would not need any "navigation" mechanisms. The Processing-agents and the Unloader-agents spread their ability information upstream using passive information sharing (Section 4.2).

If information is shared passively, it is outside of the control of the provider how the information is adapted to the local context. In general, there is no adaptation at all, as it is the case in the attraction of Focus-agents to View-agents (Section 5.2). But the propagation may also be restricted to the context into which it is put. Such a restriction is set for the $P_A$ pheromones that do not propagate beyond the next upstream processing resource (Section 4.2.1). Finally, propagation filters may be specified to automatically translate information from one context to another. The *GMC* system defines very complex filters for the translation of advice (Section 4.4.2).

An agent may ignore the propagation mechanisms of the *PI* and spread its information actively. In this case, the agent remains in control of what information is put where. Active global information sharing is more selective and dynamic, because the reasoning process may incorporate the current situation and the current local context.

Active information sharing generates the predictive pattern of $P_P$ pheromones (Section 4.3.1). The Workpiece-agents regularly spawn new Ghost-agents, which simulate a possible future of their respective creator by emulating its transport decision process and by simulating processing activities. A refresh of a $P_P$ pheromone by a Ghost-agent includes its current simulated processing state and its location, which is a result of the emulated transport decisions. It would be close to impossible to create the same predictive pattern through passive information sharing with propagation filters. The $P_P$ pheromones would have to carry much more data and the propagation filters would be very complex. As a consequence, the effort spent in implementation, stabilization, and maintenance of the prediction mechanism would be much higher.

Finally, it remains to consider under what conditions direct agent interaction may be permissible or even required. When should the designer refrain from using pheromones? In general, direct communication should not be considered for global information sharing, except when the topology of the *PI* does not fit the interaction, when the data exchange might overload the processing or communication capabilities of the *PI*, or when the information cannot be encoded in pheromones. But, at least in the first case a redesign of the layout of the *PI* may still be a better alternative.

In local information sharing, when the provider and the consumer are at the same place and operate in the same time-scale, direct communication may be chosen for very simple interactions in which no aggregation of data from different sources is required. Also, joint decision-making resulting in a commitment by the agents might be better achieved in direct negotiations. Finally, when the speed of the interaction is critical for the correct operation of the whole system, the delay imposed by transmissions through the *PI* may be too long.

## 6.3 The GMC System in State Space

The following section re-visits the domain of manufacturing control systems. Taking a step back, the operation of a manufacturing system in state space is considered. Based on the abstraction, reasoning modes available to a control system are sketched and referred to

the *GMC* system.

### 6.3.1 Manufacturing Control in State Space

#### Parameters, State Space, and Trajectories

A manufacturing system, like other systems, has a set of system parameters, which influence the behavior of the system. Some of these parameters may be controlled by the attached manufacturing control system (e.g., material flow patterns, processing capacities). Other system parameters elude direct control (e.g., processing yield, availability of resources).

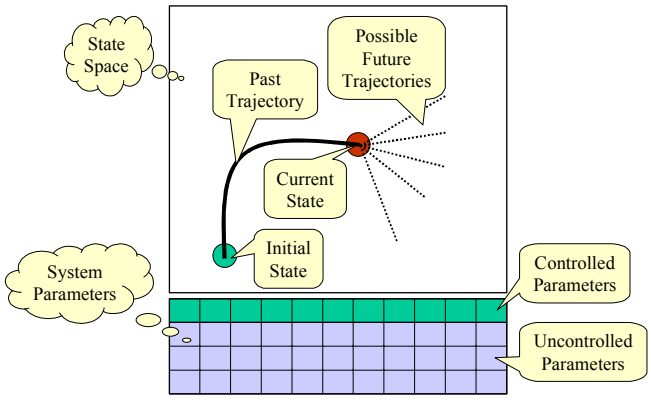


Figure 6.1. A System in State Space

There are system variables (e.g., buffer-levels, WIP, processing states of workpieces, etc.) whose respective domains are combined to span a state space of the (manufacturing) system. A setting of these system variables defines a state of the system. The manufacturing system has an initial state the moment it is started and it has a current state at some observation point in time (Figure 6.1).

The behavior of the manufacturing system as a whole follows its intrinsic dynamics (e.g., transport and processing of workpieces), which is influenced by the system parameters. As a result of the dynamics, the system variables change over time. Models in systems theory specify these intrinsic dynamics either in discrete transition functions or in continuous flow fields.

The path of the changing system variables over time in state space is called the trajectory of the system. For any moment at which the state is observed, there is exactly one trajectory starting at the initial state and ending at the current state. From the current moment on, there may be an infinite number of possible future trajectories the system may follow. The selection of one of these trajectories depends on control decisions and on changes and disturbances influencing the system parameters.

#### Performance-Evaluation

At any point in time there exists an evaluation of the previous operation of the manufacturing system. Such an evaluation may be one-dimensional even when there are multiple production goals, because these goals have to be weighted in importance and thus

they may be combined into one value for an evaluation.

An evaluation of the performance of a manufacturing system usually considers the fulfillment of maintenance goals. In this case, an evaluation may be based on a sequence of states over time (trajectory) instead of just evaluating the final state, as it is sufficient for achievement goals. Hence, the performance evaluation of a manufacturing system may be seen as a mapping from the space of system trajectories to the space of performance values. Usually only limited trajectories of a fixed length are considered. An example for an evaluation that is based on a limited trajectory is the throughput of a manufacturing system over a given period.

Consider a space of possible trajectories and its mapping to evaluations. It is the going concern of any control system to keep the evolving trajectory of a manufacturing system in a sub-space of the trajectory space where evaluations are high. Good control decisions reduce the set of future trajectories; cutting off sub-spaces of inferior evaluation and increasing the chance to access highly evaluated trajectories even though unforeseen events may occur.

### Structures in State Space

A state space may have attractors, each with a basin of attraction. If the current state of the system falls into such a basin and random events do not push it out of the basin again, the intrinsic dynamics of the manufacturing system eventually take the state into the attractor (or arbitrarily close to it). Depending on the type of attractor, the state of the system does not change anymore (point attractor), it repeats itself infinitely (limit cycle attractor), or it runs on quasi-periodic or strange orbits.

How the intrinsic dynamics of the system change the system variables (and thus the state) depends on the current setting of the system parameters. As a consequence, the location, the size and the "depth" of the basin, and even the type of an attractor may change when the system parameters change.

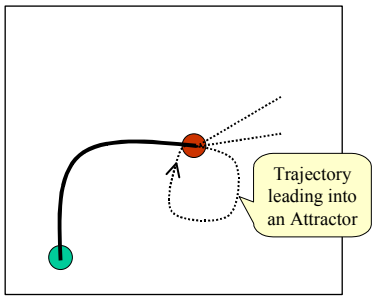


Figure 6.2. Predicting an Attractor in State Space

What is the advantage of considering attractors in state space? If the current state is located in the basin of attraction, the control system may approximate the continued trajectory of the manufacturing system, which approaches the attractor following its own intrinsic dynamics. Thus, the control system is able to decide if it has to interfere or if it can leave the system to its own dynamics for a while, assuming the uncontrolled system parameters do not change the attractor. Figure 6.2 shows a scenario where at least one of the future possible trajectories leads into a limit cycle type of attractor.

A good control system should prefer attractors with many highly evaluated incoming

trajectories that start at the current state of the system. Additionally, the basin of attraction of these attractors should be "near" the current state of the system in terms of control effort required to get into the basin. The basin should be large and deep too. Finally, the best of all attractors fulfilling the previous requirements are those that remain reasonably stable in the face of the most probable changes in the system parameters.

### Levels of Sophistication in Control

The decision processes of the control system may operate on different levels of sophistication incorporating information of different complexity. In general, there may be four levels of sophistication identified: reactive, retrospective, predictive, and proactive decisions. All four levels of sophistication are found in the *GMC* system.

On the level of reactive control, every decision is based only on the current state of the manufacturing system. The reactive level of sophistication is already sufficient for the control of a manufacturing system if the set of trajectories remaining after a control decision was taken may be determined directly from the current state. Such a prediction may be based on heuristics. In the *GMC* system, the reactive layer implements this mode of reasoning. But, in a flexible flow shop, the reactive mode is insufficient to fulfill the production goals.

In the retrospective mode the reasoning is extended to incorporate past states and the trajectory leading up to the current state. At the retrospective level, system parameters defined over a period of time, such as local throughput values, are incorporated into the decision processes too. The *GMC* system realizes the retrospective level of reasoning in the advisory system. Information on the previous evolution of the system is encoded into the current pattern created by the interface layer. The strategy implementation layer and the strategy selection mechanisms incorporate information from the past and the present into their decision processes.

Whereas the retrospective mode considers the actual trajectory in the past, predictive reasoning explores the short-term effect of alternatives in the decision process. With the consequences of a decision made explicit, the appropriate sub-set of the available trajectories is selected. The predictive way of reasoning is found in the strategy implementation layer of the *GMC* system. A short-term prediction of the evolution of the system is generated under the assumption of continued operation (stable yield and advice) by the interface layer. The strategy implementation layer takes a prediction into account when tuning the advice.

Finally, on the proactive level the control system explicitly or implicitly tries to guide the state of the manufacturing system towards an area in trajectory space where most of the trajectories map to high evaluations and where pathologic states are unlikely. In such "optimal" areas even the effects of many possible changes and disturbances are not as devastating for the performance of the system as they may be in other areas. Learning processes in the strategy evaluation and generation layer bias the *GMC* system towards strategies that are successful in terms of the evaluation of the performance of the system. The resulting trend implements an implicit kind of proactiveness. Explicit proactive reasoning is not realized.

### 6.3.2 Observing the State Space

The control system must be enabled to actually perceive the state space and important system parameters to reason about previous or possible trajectories. There are at least two ways of accessing the required data: Either, the operation of a manufacturing system is monitored directly, or the trajectory is derived from pheromones inside the control system.

Direct monitoring is primarily the task of agents at the core of the control system that map to the physical entities in the manufacturing system. For instance, a Switch-agent could monitor the arrival pattern of workpieces at its entries. Observation activities amount to an additional task besides the fulfillment of routing requests. If agents in other components of the control system need access to the gathered information, the data has to be communicated indirectly. The use of direct communication would violate the design principles.

A direct monitoring of the state of the system has the advantage that it provides up-to-date and correct data. A disadvantage is the additional task given to the core part of the control system. But, if the decision processes inside the control system actually incorporate the trajectory information into their reasoning, it might be worthwhile to spend the additional effort in design and implementation.

On the other hand, the indirect approach might be considered if trajectory information is only required for observation purposes, for instance to study the general behavior of the system. Then, the task of information gathering should be given to the visualization system whose agents have access to all pheromones at all places in the control system. The conditions for deriving the characteristics of a trajectory from pheromones are considered in the following.

The *GMC* system applied in the paint-shop demonstration (Section 4.5) illustrates the observation problem. At the place of the Processing-agent "P1" there are always a number of Workpiece-agents present. These agents regularly refresh one of the $P_S$ pheromones. Thus, in the pheromone strength the number of workpieces of a given processing state is transmitted to every agent located at "P1".

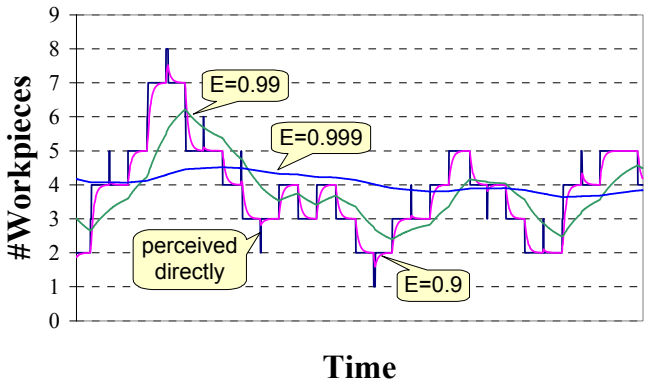


Figure 6.3. Perceived Workpiece Count at Place "P1" over Time

Depending on the evaporation parameter of $P_S$, the pheromone strength takes a certain time to approximate the correct value. Therefore, the strength conveys a short-term average of the number of workpieces present. Figure 6.3 shows the averaging effect where one curve represents the actual number of workpieces present while the plots "E=0.9",

“E=0.99”, and “E=0.999” represent the workpiece count as perceived through $P_S$ for the respective evaporation parameter $E$.

Takens's Theorem [Takens, 1980] may provide an approach to analyze the characteristics of the trajectory of the manufacturing system. Informally, it states that if there is a time series generated from the observation of a state variable, a synthetic n-dimensional state vector may be constructed under specific conditions using successive n-tuples of the time series that is projected onto a n-dimensional sub-space but still captures the topology of the trajectory of the system in its native state space completely. One of the requirements set by the theorem is the existence of the second derivative of the state variable. It may be speculated that this requirement is more easily fulfilled in the indirect observation of the variable through some related pheromones.

In the demonstration, the number of workpieces present may be observed directly by observing the number of Workpiece-agents located at place “P1”. Or, it could be observed indirectly through the respective $P_S$ pheromone. If the indirect approach is chosen, it has to be guaranteed that the time series generated through the observation of the pheromone strength is strongly related to the actual workpiece count at “P1”. Thus, the sampling rate must be set in a way that most samples are taken when the pheromone strength is in equilibrium. Based on results gained in the analysis of the *PI*, the appropriate sampling rate may be selected when estimating the rate of change of the observed variables.

In specific scenarios an estimation of the quality of a time series generated from the observation of the pheromone strength may be given. For example, in the case of the number of workpieces at “P1” it may be assumed that the workpiece count changes on average every $t_C$ time units by one workpiece. If a distance to the equilibrium of $x$ percent of the average refresh per unit time is considered sufficiently close to the equilibrium, equation (3.29) may be used to determine if (on average) the equilibrium is reached at all. Hence, the time to equilibrium $t_E$ must be smaller than $t_C$. Furthermore, the probability of sampling an equilibrium value is $(t_C–t_E)/t_C$ (percent).

# Chapter 7 - Conclusion

*The concluding chapter returns to the four questions raised in the introduction of the thesis (Chapter 1). It summarizes the presented results, evaluates how they succeed in answering the questions, and it points the way to future research.*

## 7.1 Summary

Consider again the initially raised problem of creating batches by local routing activities in a distributed transport system. The solution to the problem presented in Section 1.1.1 is the result of an empirical design process. This thesis presents an extensive set of design guidelines to enable a systematic design of synthetic ecosystems when combined with a general agent-system design method as for instance in [Burmeister, 1996].

Do the guidelines of Section 3.1 match the empirically designed successful solution of Section 1.1.1? The Router-agents represent physical entities, they are small in size and impact, they sense and act locally, they interact indirectly, their decisions contain a random element, and the control of the transport system and the fulfillment of the batching goal emerges from bottom up. Many important principles are covered by the intuitive solution already. Other principles, as for instance the suggested diversity of agents or the clustering of the agent system, are not included because the selected application is too small.

The functionality and applicability of the self-organized batching solution is extended when the workpieces are moved according to their specific routing goals. The Router-agent behavior as it is specified in Section 1.1.1 requires an open and directed layout. At each local routing point every workpiece arriving at any of the entries may be taken to any of the exits. If the layout is not open, some exits of a router may be explicitly forbidden for some workpieces. If directedness cannot be guaranteed, local routing has to differentiate between globally forward and globally backward directions in some way in order to prefer the forward direction on average. To include directed routing, more information has to be provided and incorporated into the decision processes of the agents.

A general solution to the directed routing problem follows the approach taken in the reactive layer of the *GMC* system (Section 4.2). The agents of the control system are embedded in the pheromone infrastructure. There is a separate place for each Router-agent, and there is an upstream or downstream link to another place for each entry or exit linking the router to other transport units. The advantage of including sign-based stigmergy into the agent coordination is that global information is made locally available in respect to the local context. In the case of the routing problem, the global information concerns the paths from the current position to the exits of the segment. The global information is translated into a local one that provides guidance in reference to the different local exits.

Assume that there are pheromones perceived separate for each downstream exit of a place of a Router-agent. The different pheromone types indicate the different global exits available to the workpieces that leave the system. Therefore, if a pheromone concentration larger than zero is perceived for a local exit, a step in such a direction leads towards the global exit indicated by the pheromone. Assume further that of any two local exits the one with the stronger concentration of a pheromone is part of the shorter path to the global exit. Hence, the global routing problem may be solved by local hill climbing alone.

To combine the sign-based stigmergy of the routing mechanism with the sematectonic stigmergy coordinating the creation of batches, every workpiece is assigned an agent. The task of a Workpiece-agent is to make sure its workpiece eventually reaches the correct global exit. To that end it interacts with the local Router-agent that currently handles its workpiece, and tells it the preferred local exit. The additional information reduces the choice of the Router-agent when selecting the next transport. If there was an exit *Y* that matches the product parameter value of the workpiece but at the same time *Y* is currently not preferred by the Workpiece-agent, the Router-agent is not permitted to send the workpiece to exit *Y*. With the creation of the Workpiece-agent the agent system diversifies.

There is a performance tradeoff between a direct routing of the workpieces and the creation of batches. Batching requires a delay of some workpieces while others overtake them on alternative paths. Hence, a Workpiece-agent has to permit the Router-agent to take the workpiece on a less optimal path as long as the path still leads the workpiece to the global goal. Introducing randomization in the outcome of a decision of an agent, the Workpiece-agent selects the preferred local exit probabilistically from the set of exits that offer a path to the correct global exit. The probability for an exit to be selected is linked to the relative strength of the pheromone concentration of the exit. The more dominating the pheromone concentration of one exit is, the more probable is its selection. The decision is also randomized in time. There is a timeout set for the preference choice of the Workpiece-agent after which the agent again has to select an exit. The explicit randomization prevents deadlocks and eventually covers all available routing alternatives.

The extension of the control of a transport system as it is sketched here again underlines the importance and usefulness of the design principles stated in this thesis. These principles are motivated in a general argument in Section 3.1 and on the basis of the *GMC* system in Section 6.1. The systems considered in respect to the design principles all show global properties such as robustness, adaptivity, or scalability. The first question (**"Design"**) stated in the goals of the thesis in Section 1.1.2 is answered.

Future research may change the set of design principles adding newfound principles or removing those that prove themselves not founded enough. But in addition, research effort must be invested into quantitative constraints of self-organization. Particularly, the effectiveness of stigmergetic coordination strongly depends on the number of individuals participating. If there are too less or too many agents, coordination may fail. As there are now qualitative design principles, stating what the agents are, how they interact, or what the system architecture should look like, there must be rules supporting the design and tuning of coordination mechanisms on the basis of the number of participating agents or the quantitative dynamics of the environment.

Section 3.3 answers the second question (**"Realization"**) concerning the support for the implementation of stigmergetic agent coordination mechanisms in software systems. There, a specification for the implementation of the pheromone infrastructure in an agent runtime environment is given. The pheromone infrastructure extends the services of a runtime environment and provides the agents of an application with generic services required to implement sign-based stigmergetic coordination.

The formal model of the pheromone infrastructure (Section 3.2) aims at the third question (**"Evaluation"**) this thesis should answer. In general scenarios in Section 3.2 and specifically in the discussion of the emerging behavior of the *GMC* system in Section 4.5 the formal model is used to predict, tune, and evaluate specific agent interactions and pheromone-based coordination mechanisms. Finally, it is argued in Section 6.2 that such a bottom up approach may eventually lead to the formal evaluation of emerging agent

system behavior.

The fourth question (**"Application"**) is different from the other three. It states an engineering problem that is to be solved applying the design principles, the pheromone infrastructure, and its formal model. The *GMC* system presented in Chapter 4 is a manufacturing control system that combines robustness and flexibility with optimization on the basis of production goals. Three components of the system (control system, interface layer, advisory system) are specified in detail (Sections 4.2-4.4) and their emerging behavior is evaluated in a small but realistic example of a manufacturing system (Section 4.5).

The actual optimization process runs outside of these three layers. Section 5.1 proposes concepts that should realize an automatic on-line optimization of the material flow in the manufacturing system according to the production goals. These concepts have yet to be translated into detailed specifications before they may be evaluated in the same way the lower layers are evaluated.

Another path for future research is opened up in Section 5.2. Distributed self-organizing manufacturing systems require new approaches to visualization. The visualization methods should intuitively integrate into the control system. Robustness, adaptability, self-organization, and scalability are requirements not only for the control system itself, but also for the visualization. The architecture of the *GMC* system provides a basis to follow a new approach to visualization too.

## 7.2 Future Research II

In "Future Research I" (Chapter 5), concepts for the further extension of the *GMC* system are presented in detail. The following section suggests additional research activities. It proposes the exploration of quantitative principles for the design of synthetic ecosystems, an extension of the *PI*, and the improvement of the support for tuning and evaluation of the system-level features that emerge.

### 7.2.1 Quantitative Design Principles

Chapter 3 proposes a number of qualitative design principles for synthetic ecosystems. It is a qualitative characteristic of the agents to be small and heterogeneous, to share knowledge, or to sense and act locally. Quantitative characteristics of the resulting agent system are only an effect of the application of these qualitative principles. For instance, if the agents are small, simple, and localized and if the control is emergent, then the resulting system often comprises a large number of agents. But, the designer still does not know what granularity to choose, and hence the actual number of agents and the employed coordination mechanisms may vary widely for the same application problem.

Future research into design principles should focus on quantitative support for the designer. The emergence of multi-agent coordination depends on the number of agents participating. For every chosen coordination mechanism there may be a minimum and a maximum number of agents beyond which the intended system-level behavior fails to appear. If there are not enough agents, then the influence of the single individual is too strong and the required statistical or probabilistic abstraction is not given (Section 6.2.1). On the other hand, if there are too many agents, important differences in the states of the

system may not be perceived anymore because they all "blur" into one or two major states. Furthermore, too many agents at one place may overload the local server.

Quantitative design principles do not only refer to the number of agents, they should also support the setup of the dynamics of the system. For instance, the response time is an important feature in real-world application systems. The numerical analysis of the *PI* includes a prediction of the time required for pheromone patterns to stabilize in a given scenario (Section 3.2). Such results should be extended to characterize specific emergent coordination mechanisms.

### 7.2.2 Extending the Pheromone Infrastructure

The *PI* is a generic extension of runtime environments for software agents that enables the agents of the application to sense pheromone patterns and to place pheromones in arbitrary quantities. Neither the infrastructure, nor its formal model implies a specific input behavior of the agents. Any input pattern in space and time is permitted.

The analysis of the *PI* focuses on a specific local input pattern generated by a repeated input of a fixed amount of a pheromone (Section 3.2). A regular input pattern is the basic assumption for a number of numerical predictions of pheromone patterns and it occurs in the *GMC* system many times.

An extension of the *PI* may provide specific support to the agents for their regular refresh. An application agent might want to register the parameters of the regular refresh (input strength, input rate) instead of regularly sending input messages to its current Place-agent. In turn the Place-agent registers a weaker regular refresh with its neighbors in the propagation direction of the pheromone. The registration is spread recursively as long as the registered input strength has not fallen below a fixed threshold. If the application agent decides to stop the input, it de-registers the regular refresh.

The "emulation" of regular refresh activities by the network of Place-agents requires a change in the local pheromone management. In the implementation proposed in Section 3.3, a Place-agent computes the current pheromone strength whenever a new input occurs or when an agent accesses the pheromone. Therefore, the Place-agent only has to know the time that has passed since the last update and it just applies the simple evaporation function $s(t+\Delta t) = s(t) * E^{\Delta t}$. But, the new transition function is much more complex if the Place-agent should compute the new pheromone strength asynchronously, incorporating all emulated regular refresh actions that would have occurred since the last update.

A regular refresh activity is not intended to provide its inputs at specific points in time. It rather sets off the evaporation to some degree. Such an interpretation already permitted a significant reduction of the complexity of the numerical predictions in Section 3.2 and it may also reduce the complexity of the new transition function when regular refresh activities are emulated. The Place-agent "just" determines the offset to the evaporation parameter, which results from the currently registered regular refresh activities.

The statistical emulation of regular refresh activities has a positive sideffect. The pheromone strength oscillates as long as the application agents provide the input themselves, especially when the parameters of the refresh behavior are set to widely spaced strong inputs. As a consequence, the error in sampling the "intended" pheromone strength may be large. In the statistical emulation the error is gone, because all refresh activities are normalized to one input per unit time. With the inherent normalization of the impact of the

single agent, the minimally required number of agents for emergence (see previous section) may actually be lower. But in any case, the communication between the application agents and the *PI* is significantly reduced.

### 7.2.3 Improving Tuning and Evaluation

The development of quantitative design principles and the proposed extension of the *PI* have consequences for the tuning and the evaluation of the emerging system-level behavior. It should be a goal to have a "theory of emergence" that formally predicts the effectiveness of a coordination mechanism in the face of a given problem by the number of agents employed. Such a theory should also tell a designer the optimal parameter settings. For example, the tradeoff between the longest path of a Ghost-agent and the quality of the emerging prediction of a future pattern in the *GMC* system should be balanced.

On the other hand, the agents could be made more adaptive. Instead of tuning every parameter of a coordination mechanism, each agent should learn the best parameter setting for the application at hand. Future research should specify a generic extension of agents that operate in the *PI* to support learning of agent parameters, and reinforcement learning may be an attractive candidate.

The *PI* has some similarities to artificial neural networks. Neural networks also specify a set of spatial locations (neurons) where activation levels are stored and there is also a propagation of activation events among these locations. The transition function in the *PI* differs from the activation function in neurons. Future research should investigate if the similarities between artificial neural networks and the *PI* are sufficient to adapt findings.

Results gained in the research into complex dynamical systems should be applied to support the designer in the tuning and evaluation process. The stability of emergent system-level features in the face of external changes and disturbances is an important issue in the evaluation of an application system. Self-organization often includes multi-stability and the behavior of the interacting individuals may be non-linear. The stable state of the system is identified with a number of features that are observed. The stable states of a given application system and their respective features are determined in the evaluation as well as the requirements for transitions from one stable state to another.

# Bibliography

[Akkiraju et al., 1998] Rama Akkiraju, Pinar Keskinocak, Sesh Murthy, and Frederick Wu. Multi machine scheduling: An agent-based approach. In Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98) and of the 10th Conference on Innovative Applications of Artificial Intelligence (IAAI-98), pages 1013-1019, Menlo Park, July 26-30 1998. AAAI Press.

[Andersson and Sandholm, 1998] Martin R. Andersson, and Thomas W. Sandholm. Leveled commitment contracts with myopic and strategic agents. In Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98) and of the 10th Conference on Innovative Applications of Artificial Intelligence (IAAI-98), pages 38-45, Menlo Park, July 26-30 1998. AAAI Press.

[Ashcroft and Mermin, 1976] W. Ashcroft and N.D. Mermin. Solid State Physics. W.B. Saunders, Philadelphia, 1976.

[Baker, 1996] A.D. Baker. Metaphor or reality: A case study where agents bid with actual costs to schedule a factory. In S.H. Clearwater, editor, Market-Based Control: A Paradigm for Distributed Resource Allocation, pages 184-223. World Scientific Publishing Co. Pte. Ltd., 1996.

[Batty, 1997] M. Batty. Predicting where we walk. Nature, 388:19-20, July 1997.

[Baumgärtel, 1999] H. Baumgärtel. Verteiltes Lösen von Constraint-Problemen in Multiagenten-Systemen zur optimierten Planung in einer Fließfertigung. PhD thesis, Technical University Berlin, 1999.

[Beckers et al., 1992] R. Beckers, J.L. Deneubourg, and S. Goss. Trails and u-turns in the selection of the shortest path by the ant lasius niger. Journal of theoretical biology, 159:397-415, 1992.

[Begon et al., 1996] M. Begon, D.J. Thompshon, and M. Mortimer. Population Ecology: A Unified Study of Animals and Plants. Blackwell Science Inc., 1996.

[Bennett and Grinstein, 1985] C. Bennett and G. Grinstein. Role of irreversibility in stabilizing complex and nonenergodic behavior in locally interacting discrete systems. Physical Review Letters, 55:657-660, 1985.

[Berlekamp et al., 1982] E.R. Berlekamp, J.H. Conway, and R.K. Guy. Winning ways for your mathematical plays. Academic Press, New York, 1982.

[Bonabeau et al., 1999] E. Bonabeau, M. Dorigo, and G. Theraulaz. Swarm Intelligence: From Natural to Artificial Systems. Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Press, 1999.

[Bond and Gasser, 1988] Alan H. Bond and Les Gasser. Readings in Distributed Artificial Intelligence. Morgan Kaufmann, San Mateo, 1988.

[Bongaerts, 1998] L. Bongaerts. Integration of Scheduling and Control in Holonic Manufacturing Systems. PhD thesis, K.U.Leuven, PMA Division, 1998.

[Bradshaw, 1997] Jeffrey M. Bradshaw. Software Agents. AAAI Press, Menlo Park, CA, USA, 1997.

[Breyer et al., 1998] Jens Breyer, Jörg Ackermann, and John McCaskill. Evolving reaction-diffusion ecosystems with self-assembling structures in thin films. In Christoph Adami, Richard K. Belew, Hiroaki Kitano, and Charles Taylor, editors, Proceedings of the 6th International Conference on Artificial Life (ALIFE-98), pages 28-34, Cambridge, MA, USA, June 27-29 1998. MIT Press.

[Brooks, 1991a] Rodney A. Brooks. Intelligence without reason. In Ray Myopoulos, John Reiter, editor, Proceedings of the 12th International Joint Conference on Artificial Intelligence, pages 569-595, Sydney, Australia, August 1991. Morgan Kaufmann.

[Brooks, 1991b] Rodney A. Brooks. Intelligence without representation. Artificial Intelligence, 47:139-159, 1991.

[Brueckner et al., 1998] S. Brueckner, J. Wyns, P. Peeters, and M. Kollingbaum. Designing Agents for Manufacturing Control. In Proceedings of the 2nd AI & Manufacturing Research Planning Workshop, Albuquerque, NM, August 1998.

[Brueckner, 1997] S. Brueckner. Autonome und Kooperative Auftragsdurchsetzung für die Steuerung exibler Fertigungsanlagen. Master's thesis, Humboldt University Berlin, February 1997.

[Brueckner, 1998] S. Brueckner. Manufacturing Control Systems Capable of Managing Production Change and Disturbances – An Application of Autonomous Cooperating Agents. The European Network of Excellence for Agent-Based Computing (AgentLink) Newsletter, 1, October 1998.

[Brueckner, 1999] S. Brueckner. Return from the Ant: Synthetic Ecosystems for Manufacturing Control. The European Network of Excellence for Agent-Based Computing (AgentLink) Newsletter, 4, November 1999.

[Bullnheimer et al., 1999] B. Bullnheimer, R.F. Hartl, and C. Strauss. A new rank based version of the ant system: A computational study. Central European Journal for Operations Research and Economics, 7(1):25-38, 1999.

[Burke and Prosser, 1994] P. Burke and P. Prosser. The distributed autonomous scheduler. In M.B. Morgan, editor, Intelligent Scheduling, pages 309-339. Morgan Kaufman Publishers, Inc., San Francisco, 1994.

[Burks, 1970] Arthur W. Burks. Essays on Cellular Automata. University of Illinois Press, Urbana, IL, USA, 1970.

[Burmeister, 1996] Birgit Burmeister. Models and methodology for agent-oriented analysis and design. In K. Fischer, editor, Working Notes of the KI'96 Workshop on Agent-Oriented Programming and Distributed Systems, 1996.

[Bussmann and McFarlane, 1999] S. Bussmann and D. McFarlane. Rationales for holonic manufacturing control. In Proceedings of the Second International Workshop on Intelligent Manufacturing Systems, pages 177-184, Leuven, Belgium, 1999.

[Bussmann and Schild, 2000] S. Bussmann and K. Schild. Self-organizing manufacturing control: An industrial application of agent technology. In The Fourth International Conference on Multi-Agent Systems (ICMAS'2000), 2000.

[Bussmann, 1998] S. Bussmann. An agent-oriented architecture for holonic manufacturing control. In Proceedings of the First International Workshop on Intelligent Manufacturing Systems, pages 1-12, EPFL, Lausanne, Switzerland, 1998.

[Butler and Ohtsubo, 1992] J. Butler and H. Ohtsubo. ADDYMS: Architecture for distributed dynamic manufacturing scheduling. In A. Famili, D.S.Nau, and S.H. Kim, editors, Artificial Intelligence Applications in Manufacturing, pages 199-214. AAAI Press/The MIT Press, Menlo Park, CA, 1992.

[Christensen, 1994] J. Christensen. Holonic manufacturing systems - initial architecture and standards directions. In Proceedings of First European Conference on Holonic Manufacturing Systems, Hannover, Germany, 1994.

[Colorni et al., 1993] A. Colorni, M. Dorigo, F. Maffioli, V. Maniezzo, G. Righini, and M. Trubian. Heuristics from nature for hard combinatorial problems. In Proceedings of IFORS 93 -XIII World Conference on Operations Research, Lisbon, Portugal, July 12-16 1993.

[Cook and Racko, 1980] Stephen A. Cook and Charles W. Racko. Space lower bounds for maze threadability on restricted machines. SIAM Journal on Computing, 9(3):636-652, 1980.

[Deamer and Oro, 1980] D. W. Deamer and J. Oro. The role of lipids in prebiotic structures. Biosystems, 12:167-175, 1980.

[Delgado and Sole, 1997] Jordi Delgado and Ricard V. Sole. Collective-induced computation. Physical Review E, 55:2338-2344, 1997.

[Deneubourg et al., 1989] J. L. Deneubourg, S. Gross, N. Franks, and J. M. Pasteels. The blind leading the blind: Modeling chemically mediated army ant raid patterns. Journal of Insect Behavior, 2:719-725, 1989.

[Dewan and Joshi, 1998] P. Dewan and S. Joshi. Distributed control of manufacturing systems with due date constraints. In Proceedings of IECR, 1998.

[Di Caro and Dorigo, 1998a] G. Di Caro and M. Dorigo. Antnet: Distributed stigmergetic control for communications networks. Journal of Artificial Intelligence Research (JAIR), 9:317-365, 1998.

[Di Caro and Dorigo, 1998b] G. Di Caro and M. Dorigo. Mobile agents for adaptive routing. In 31st Hawaii International Conference on System Science, Big Island of Hawaii, January 1998.

[Dorigo and Di Caro, 1999] M. Dorigo and G. Di Caro. The ant colony optimization metaheuristic. In D. Corne, M. Dorigo, and F. Glover, editors, New Ideas in Optimization. McGraw-Hill, 1999.

[Dorigo and Gambardella, 1997] Marco Dorigo and Luca Maria Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, 1(1):53-66, April 1997.

[Dorigo et al., 1996] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. The Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics, 26(1):29-41, 1996.

[Dorigo et al., 1999] M. Dorigo, G. Di Caro, and L.M. Gambardella. Ant algorithms for discrete optimization. Artificial Life, 5(2), 1999.

[Dorigo, URL] M. Dorigo. Behavior of real ants. In Ant Colony Optimization Homepage. http://iridia.ulb.ac.be/~mdorigo/ACO/RealAnts.html.

[Drogoul, 1993] Alexis Drogoul. When Ants Play Chess. In C Castelfranchi and J.P. Müller, editors, From Reaction to Cognition, pages 13-27. Springer Verlag, LNAI 957, 1993.

[Eberhart and Shi, 1998] R. C. Eberhart and Y. Shi. Comparison between genetic algorithms and particle swarm optimization. Lecture Notes in Computer Science, 1447:611, 1998.

[Edwards and Peng, 1998] Linglan Edwards and Yun Peng. Computational models for the formation of protocell structures. In Christoph Adami, Richard K. Belew, Hiroaki Kitano, and Charles Taylor, editors, Proceedings of the 6th International Conference on Artificial Life (ALIFE-98), pages 35-42, Cambridge, MA, USA, June 27-29 1998. MIT Press.

[Eliasmith, URL] Chris Eliasmith. Dictionary of philosophy of mind. Washington University in St.Luis. http://artsci.wustl.edu/~philos/MindDict.

[Epstein and Axtell, 1996] J. Epstein and R. Axtell. Growing artificial societies: Social science from the bottom up. 1996.

[Ermentrout and Edelstein-Keshet, 1993] G. Bard Ermentrout and Leah Edelstein-Keshet. Cellular automata approaches to biological modeling. Journal of Theoretical Biology, 160:97-133, January 1993.

[Ferguson et al., 1988] Donald Ferguson, Yechiam Yemini, and Christos Nikolaou. Microeconomic Algorithms for Load Balancing in Distributed Computer Systems. In International Conference on Distributed Computer Systems, pages 491-499, 1988.

[Flake, 1999] G.W. Flake. The Computational Beauty of Nature: Computer Explorations of Fractals, Chaos, Complex Systems, and Adaptation. The MIT Press, 1999.

[Fordyce and Sullivan, 1994] K. Fordyce and G.G. Sullivan. Logistics management system (lms): Integrating decision technologies for dispatch scheduling in semiconductor manufacturing. In M.B. Morgan, editor, Intelligent Scheduling, pages 473-516. Morgan Kaufman Publishers, Inc., San Francisco, 1994.

[Forgy, 1982] C. L. Forgy. Rete: A fast algorithm for the many pattern/many object pattern matching problem. Artificial Intelligence, 19:17-37, 1982.

[Forrest et al., 1994] Stephanie Forrest, Alan S. Perelson, Lawrence Allen, and Rajesh Cherukuri. Self-nonself discrimination. In Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy, pages 202-212. IEEE Computer Society, IEEE Computer Society Press, May 1994.

[Fox et al., 1993] M.S. Fox, J.F. Chionglo, and M. Barbuceanu. The integrated supply chain management system. Technical report, Department of Industrial Engineering, University of Toronto, Toronto, Ontario, 1993.

[Franklin and Graesser, 1997] S. Franklin and A. Graesser. Is it an agent, or just a program? A taxonomy for autonomous agents. Lecture Notes in Computer Science, 1193, 1997.

[Gabora, December 1998] Gabora. Autocatalytic closure in a cognitive system: A tentative scenario for the origin of culture. Psycology, 9(67), December 1998.

[Gambardella et al., 1999] L.M. Gambardella, E. Taillard, and M. Dorigo. Ant colonies for

the quadratic assignment problem. Journal of the Operational Research Society, 50:167-176, 1999.

[Gardener, 1970] M. Gardener. The fantastic combinations of john conway's new solitaire game „life“. Scientific American, pages 120-123, October 1970.

[Gardener, 1971] M. Gardener. On cellular automata, self-reproduction, the garden of eden and the game „life“. Scientific American, 224(2):112-117, Februar 1971.

[Goss et al., 1989] S. Goss, S. Aron, J.L. Deneubourg, and J.M. Pasteels. Self-organized shortcuts in the argentine ant. Naturwissenschaften, 76:579-581, 1989.

[Grassè, 1959] P.-P. Grassè. La reconstruction du nid et les coordinations inter-individuelles chez bellicositermes natalensis et cubitermes sp. la thèorie de la stigmergie: Essai d'interprètation du comportement des termites constructeurs. Insectes Sociaux, 6:41-80, 1959.

[Grassè, 1984] P.-P. Grassè. Termitologia, Tome II. Fondation des Sociètès. Construction. Masson, Paris, 1984.

[Haken, 1983] H. Haken. Synergetics. An Introduction. Springer, Berlin, 1983.

[Hanski et al., 1996] I. Hanski, A. Moilanen, and M. Gyllenberg. Minimum viable population size. American Naturalist, 147:527-541, 1996.

[Hastings, 1997] A. Hastings. Population Biology: Concepts and Models. Springer-Verlag, 1997.

[Helbing et al., 1997a] D. Helbing, J. Keltsch, and P. Molnar. Modeling the evolution of human trail systems. Nature, 388:47-49, July 1997.

[Helbing et al., 1997b] D. Helbing, F. Schweitzer, J. Keltsch, and P. Molnar. Active walker model for the formation of human and animal trail systems. Physical Review E, 56(3):2527-2539, 1997.

[Hofmeyr and Forrest, 1999] Steven A. Hofmeyr and Stephanie Forrest. Immunity by design: An artificial immune system. In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela, and Robert E. Smith, editors, Proceedings of the Genetic and Evolutionary Computation Conference, volume 2, pages 1289-1296, Orlando, Florida, USA, 13-17 July 1999. Morgan Kaufmann.

[Hölldobler and Wilson, 1990] B. Hölldobler and E.O. Wilson. The ants. Springer-Verlag, Berlin, 1990.

[Huberman and Hogg, 1988] Bernardo A. Huberman and Tad Hogg. The behavior of computational ecologies. In B. A. Huberman, editor, The Ecology of Computation, pages 77-176. North-Holland Publishing Company, Amsterdam, 1988.

[Huhns, 1987] M. N. Huhns. Distributed Artificial Intelligence. Pitman/Morgan Kaufmann, San Mateo, CA, USA, 1987.

[Hynynen, 1989] J.E. Hynynen. Boss: An artificially intelligent system for distributed factory scheduling. In F. Kimura and A. Rolstadas, editors, Computer Applications in Production and Engineering, pages 667-677. Elsevier, 1989.

[Interrante and Goldsmith, 1998] L. Interrante and S. Goldsmith. Emergent agent-based

scheduling of manufacturing systems. In Proceedings of Workshop on Agent-Based Manufacturing, pages 47-56, Minneapolis, MN, May 1998.

[Iyer and Ghosh, 1995] R. V. Iyer and S. Ghosh. Daryn, a distributed decision-making algorithm for railway networks: Modeling and simulation. IEEE transaction of Vehicular Technology, 44(1):180-191, 1995.

[Jennings et al., 1998] N. R. Jennings, K. Sycara, and M. Wooldridge. A roadmap of agent research and development. Journal of Autonomous Agents and Multi-Agent Systems, 1(1):7-38, 1998.

[Kauffman, 1993] S. Kauffman. The Origins of Order: Self-Organization and Selection in Evolution. Oxford University Press, Inc., 1993.

[Kauffman, 1995] S. Kauffman. At Home in the Universe: The Search for Laws of Self-Organization and Complexity. Oxford University Press, Inc., 1995.

[Kelly, 1994] K. Kelly. Out of Control: The New Biology of Machines, Social Systems, and the Economic World. Perseus Books, 1994.

[Kennedy and Eberhart, 1995] J. Kennedy and R.C. Eberhart. Particle swarm optimization. In Proceedings of the 1995 IEEE International Conference on Neural Networks, volume IV, pages 1942-1948, Piscataway, NJ, 1995. IEEE Service Center.

[Kephart, 1994] Jeffrey O. Kephart. A biologically inspired immune system for computers. In Rodney A. Brooks and Pattie Maes, editors, Proceedings of the 4th International Workshop on the Synthesis and Simulation of Living Systems Artificial Life IV, pages 130-139, Cambridge, MA, USA, July 1994. MIT Press.

[Koestler, 1967] A. Koestler. The Ghost in the Machine. Hutchinson & Co, London, 1967.

[Koza, 1992] J. R. Koza. Genetic Programming. MIT Press, Cambridge, MA, 1992.

[Kraus, 1997] S. Kraus. Negotiation and cooperation in multi-agent environments. Artificial Intelligence, pages 79-97, 1997.

[Langton, 1992] C. Langton. Preface. In C. Langton et al., editor, Artificial Life II, volume X of SFI Studies in the Sciences of Complexity, pages xiii-xviii. Addison-Wesley, New York., 1992.

[Lee et al., 1996] H. S. Lee, S. S. Murthy, S. W. Haider, and D. V. Morse. Primary production scheduling at steelmaking industries. IBM Journal of Research and Development, 40(2):231-252, March 1996.

[Lee et al., 1999] T. S. Lee, S. Ghosh, J. Liu, X. Ge, and A. Nerode. A mathematical framework for asynchronous, distributed, decision-making systems with semi-autonomous entities: Algorithm synthesis, simulation, and evaluation. In Fourth International Symposium on Autonomous Decentralized Systems, Tokyo, Japan, 1999.

[Lenz and Burkhard, 1996a] Mario Lenz and Hans-Dieter Burkhard. Case Retrieval Nets: Basic Ideas and Extensions. In G. Görz and S. Hölldobler, editors, KI-96: Advances in Artificial Intelligence. Springer Verlag, LNAI 1137, 1996.

[Lenz and Burkhard, 1996b] Mario Lenz and Hans-Dieter Burkhard. Case Retrieval Nets: Foundations, Properties, Implementation, and Results. Technical report, Humboldt University Berlin, 1996.

[Lenz et al., 1996] Mario Lenz, Hans-Dieter Burkhard, and Sven Brueckner. Applying Case Retrieval Nets to Diagnostic Tasks in Technical Domains. In I. Smith and B. Faltings, editors, Advances in Case-Based Reasoning, LNAI1168. Springer Verlag, 1996.

[Lin and Solberg, 1992] G.Y.-J. Lin and J. Solberg. Integrated shop floor control using autonomous agents. IIE Transactions, 24(3):57-71, 1992.

[Malone et al., 1983] T. W Malone, R. E. Fikes, and M. T. Howard. ENTERPRISE: A market-like task-scheduler for distributed computing environments. Journal of TIMS, October 1983.

[Maniezzo and Colorni, 1999] V. Maniezzo and A. Colorni. The ant system applied to the quadratic assignment problem. IEEE Transactions on Knowledge and Data Engineering, 1999.

[Maniezzo et al., 1994] V. Maniezzo, A. Colorni, and M. Dorigo. The ant system applied to the quadratic assignment problem. Technical Report IRIDIA/94-28, Universit Libre de Bruxelles, Belgium, 1994.

[Maniezzo, 1998] V. Maniezzo. Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. Technical Report CSR 98-1, Instituto Dalle Molle di Studi sull'Intelligenza Artificiale, Università di Bologna, Sede di Cesena, Italy, 1998.

[Margolus and Toffoli, 1987] Margolus and Toffoli. Cellular automata machines. COMP-SYSTS: Complex Systems, 1, 1987.

[Marsili and Zhang, 1998] Matteo Marsili and Yi-Cheng Zhang. Stochastic dynamics in game theory. In Proceedings of the Budapest Conference ECONOPHYSICS. Kluwer, 1998.

[McFarland, 1994] D. McFarland. Toward robot cooperation. In From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior, pages 440-443. MIT Press, 1994.

[McMullin and Varela, 1999] B. McMullin and F. J. Varela. Rediscovering computational autopoiesis. Santa Fe Institute Working paper 07-02-012, 1999.

[Meyer and Brown, 1998] D.A. Meyer and T.A. Brown. Statistical mechanics of voting. Physical Review Letters, 81(8):1718-1721, 1998.

[Mill, 1843] John Stuart Mill. System of Logic. Longmans, Green, Reader, and Dyer, 1843.

[Minsky, 1986] M. Minsky. The Society of Mind. Simon and Schuster, New York, 1986.

[New and Pohorille, 1999] M.Newand and A. Pohorille. An inherited effciencies model for non-genomic evolution. In Proceedings of the 1st Conference on Modeling and Simulation in Biology, Medicine, and Biomedical Engineering, 1999.

[Nicolis and Prigogine, 1977] G. Nicolis and I. Prigogine. Self-Organization in Nonequilibrium Systems. John Wiley & Sons, New York, 1977.

[Nicolis and Prigogine, 1987] Gregoire Nicolis and Ilya Prigogine. Die Erforschung des Komplexen. Piper, Muenchen, 1 edition, 1987.

[Nishimura and Ikegami, 1997] Shin I. Nishimura and Takashi Ikegami. Emergence of collective strategies in a prey-predator game model. Artificial Life, 3:243-260, 1997.

[Odell and Greenstein, 1999] J. Odell and D. Greenstein. Agent-based manufacturing: Putting agents to work in the real world. Distributed Computing, 2(4):18-21, April 1999. www.DistributedComputing.com.

[Ozcan and Mohan, 1999] Ender Ozcan and Chilukuri K. Mohan. Particle swarm optimization: Surfing the waves. In Peter J. Angeline, Zbyszek Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, editors, Proceedings of the Congress on Evolutionary Computation, volume 3, pages 1939-1944, Mayflower Hotel, Washington D.C., USA, 6-9 July 1999. IEEE Press.

[Parunak and Brueckner, 1999] H.V.D. Parunak and S. Brueckner. Synthetic Pheromones for Distributed Motion Control. In Proceedings of DARPA-JFACC Technical Symposium on Advances in Enterprise Control. DARPA, 1999.

[Parunak and Brueckner, 2000] H.V.D. Parunak and S. Brueckner. Ant-Like Missionaries and Cannibals: Synthetic Pheromones for Distributed Motion Control. In Fourth International Conference on Autonomous Agents (Agents'2000), 2000.

[Parunak et al., 1997] H.V.D. Parunak, Albert D. Baker, and Steven J. Clark. The AARIA agent architecture: An example of requirements-driven agent-based system design. In W. Lewis Johnson and Barbara Hayes-Roth, editors, Proceedings of the First International Conference on Autonomous Agents (Agents'97), pages 482-483, New York, February 5-8, 1997. ACM Press.

[Parunak et al., 1998] H.V.D. Parunak, John Sauter, and Steve Clark. Toward the specification and design of industrial synthetic ecosystems. In Munindar P. Singh, Anand Rao, and Michael J. Wooldridge, editors, Proceedings of the 4th International Workshop on Agent Theories, Architectures, and Languages (ATAL-97), LNAI 1365, pages 45-60, Berlin, July 24-26 1998. Springer.

[Parunak, 1987] H.V.D. Parunak. Manufacturing experience with the contract net. In M.N. Huhns, editor, Distributed Artificial Intelligence, pages 285-310. Pitman, London, 1987.

[Parunak, 1997] H.V.D. Parunak. 'Go to the ant': Engineering principles from natural agent systems. Annals of Operations Research, 1997.

[Peeters et al., 1999] P. Peeters, P. Valckenaers, J. Wyns, and S. Brueckner. Manufacturing Control Algorithm and Architecture. In H. Van Brussel and P. Valckenaers, editors, Proceedings of the Second International Workshop on Intelligent Manufacturing Systems, pages 877-888, 1999.

[Perrier et al., 1996] J.-Y. Perrier, M. Sipper, and J. Zahnd. Toward a viable, self-reproducing universal computer. Physica D, 97:335-352, 1996.

[Polls, 1998] G.A. Polls. Stability is woven by complex webs. Nature, 395:744-745, 1998.

[Rabin, 1982] Michael O. Rabin. The choice coordination problem. Acta Informatica, 17(2):121-134, June 1982.

[Ramaswamy, 1995] S. Ramaswamy. Distributed Control of Automated Manufacturing Systems. PhD thesis, Penn State University, 1995.

[Reif, 1965] F. Reif. Fundamentals of Statistical and Thermal Physics. McGraw-Hill, 1965.

[Reynolds, 1987] C. W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. Computer Graphics: Proceedings of SIGGRAPH `87, 21(4):25-34, July 1987.

[Rich and Knight, 1991] Elaine Rich and Kevin Knight. Artificial Intelligence. McGraw-Hill, New York, second edition, 1991.

[Sandholm and Lesser, 1995] Tuomas W. Sandholm and Victor R. Lesser. Coalition formation amoung bounded rational agents. In Chris S. Mellish, editor, Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, pages 662-669, Montreal, Canada, August 1995.

[Sandholm et al., 1998] Tuomas Sandholm, Kate Larson, Martin Andersson, Onn Shehory, and Fernando Tohme. Anytime coalition structure generation with worst case guarantees. In Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98) and of the 10th Conference on Innovative Applications of Artificial Intelligence (IAAI-98), pages 46-53, Menlo Park, July 26-30 1998. AAAI Press.

[Schoonderwoerd et al., 1996] Ruud Schoonderwoerd, Owen E. Holland, Janet L. Bruten, and Leon J. M. Rothkrantz. Ant-based load balancing in telecommunications networks. Adaptive Behavior, 5(2):169-207, 1996.

[Schoonderwoerd et al., 1997] Ruud Schoonderwoerd, Owen Holland, and Janet Bruten. Ant-like agents for load balancing in telecommunications networks. In W. Lewis Johnson and Barbara Hayes-Roth, editors, Proceedings of the First International Conference on Autonomous Agents (Agents'97), pages 209-216, New York, February 5-8, 1997. ACM Press.

[Sen et al., 1994] Sandip Sen, Mahendra Sekaran, and John Hale. Learning to coordinate without sharing information. In Proceedings of the 12th National Conference on Artificial Intelligence. Volume 1, pages 426-431, Menlo Park, CA, USA, July 31-August 4 1994. AAAI Press.

[Shaw and Whinston, 1988] M. Shaw and A. B. Whinston. A distributed knowledge-based approach to flexible automation: The contract-net framework. International Journal of Flexible Manufacturing Systems, 1(1):85-104, 1988.

[Shehory et al., 1999] O. Shehory, S. Kraus, and O. Yadger. Emergent cooperative goal-satisfaction in large-scale automated agent systems. Artificial Intelligence, 110(1), May 1999.

[Shen and Norrie, 1998] Weiming Shen and Douglas H. Norrie. Combining mediation and bidding mechanisms for agent-based manufacturing scheduling. In Katia P. Sycara and Michael Wooldridge, editors, Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98), pages 469-470, New York, May 9-13, 1998. ACM Press.

[Shi and Eberhart, 1998] Y. Shi and R. C. Eberhart. Parameter selection in particle swarm optimization. Lecture Notes in Computer Science, 1447:591, 1998.

[Sikora and Shaw, 1997] R. Sikora and M.J. Shaw. Coordination mechanisms for multi-agent manufacturing systems: Applications to integrated manufacturing scheduling.

IEEE Transactions on Engineering Management, 44(2):175-187, May 1997.

[Smith, 1969] A. Smith. Cellular automata theory. Technical Report 2, Stanford Electronic Lab., Stanford University, 1969.

[Smith, 1992] Alvy Ray Smith. Simple nontrivial self-reproducing machines. In Christopher G. Langton, Charles Taylor, J. Doyne Farmer, and Steen Rasmussen, editors, Proceedings of the Workshop on Artificial Life (ALIFE '90), Santa Fe Institute Studies in the Sciences of Complexity, volume 5, pages 709-726, Redwood City, CA, USA, February 1992. Addison-Wesley.

[Steels, 1995] L. Steels. The homo cyber sapiens, the robot homonidus intelligens, and the ‚artificial life' approach to artificial intelligence. In Proc. Burda Symp. on Brain-Computer-Interfaces, 1995.

[Stützle and Dorigo, 1999] T. Stützle and M. Dorigo. ACO algorithms for the quadratic assignment problem. In D. Corne, M. Dorigo, and F. Glover, editors, New Ideas in Optimization. McGraw-Hill, 1999.

[Suda, 1989] Suda. Future factory system formulated in Japan. TECHNO JAPAN, 22, 1989.

[Sycara et al., 1995] K. Sycara, K. Decker, and D. Zeng. Designing a multi-agent portfolio management system. Carnegie Mellon University, Software Agents Group, 1995. http://www.cs.cmu.edu/~softagents.

[Sycara et al., 1996] K. Sycara, K. Decker, A. Pannu, M. Williamson, and D. Zeng. Distributed intelligent agents. IEEE Expert: Intelligent Systems and Their Applications, 11(6):36-46, December 1996.

[Sycara, 1998] K. Sycara. Multi-agent systems. AI Magazine, 19(2):79-92, 1998.

[Szabo and Toke, 1998] G. Szabo and C. Toke. Evolutionary prisoner's dilemma game on a square lattice. Physical Review E, 58(1):69-73, 1998.

[Taillard and Gambardella, 1997] E.D. Taillard and L. Gambardella. An ant approach for structured quadratic assignment problems. Technical Report IDSIA-22-97, Instituto Dalle Molle di Studi sull'Intelligenza Artificiale, Università di Bologna, Sede di Cesena, Italy, January 22 1997.

[Takens, 1980] F. Takens. Detecting strange attractors in turbulence. In D. A. Rand and L.-S. Young, editors, Dynamical Systems and Turbulence (Warwick 1980), volume 898 of Lecture Notes in Mathematics, pages 366-381. Springer-Verlag, Berlin, 1980.

[Tsetlin, 1973] M. L. Tsetlin. Automaton Theory and Modeling of Biological Systems. Academic Press, New York, 1973.

[Upton et al., 1991] D.M. Upton, M.M. Barash, and A.M. Matheson. Architectures and auctions in manufacturing. International Journal of Computer Integrated Manufacturing, 1:23-33, 1991.

[Upton, 1992] David M. Upton. A flexible structure for computer-controlled manufacturing systems. Manufacturing Review, 5(1):58-74, 1992.

[Vaario and Ueda, 1998] J. Vaario and K. Ueda. An emergent modeling method for dynamic scheduling. Journal of Intelligent Manufacturing, 9(2):129-140, April

1998.

[Van Brussel et al., 1998] H. Van Brussel, J. Wyns, P. Valckenaers, L. Bongaerts, and P. Peeters. Reference architecture for holonic manufacturing systems. Computers in Industry, special issue on IMS, 37(3):255-276, 1998.

[Varela and Sinclair, 1999] Griselda Navarro, Varela and Mark C. Sinclair. Ant colony optimisation for virtual-wavelength-path routing and wavelength allocation. In Peter J. Angeline, Zbyszek Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, editors, Proceedings of the Congress on Evolutionary Computation, volume 3, pages 1809-1816, Mayflower Hotel, Washington D.C., USA, 6-9 July 1999. IEEE Press.

[Vichniac, 1984] G. Y. Vichniac. Simulating physics with cellular automata. Physica D, 10:96-116, 1984.

[von Neumann, 1966] John von Neumann. Theory of Self-Reproducing Automata. University of Illinois Press, Urbana, IL, USA, 1966.

[Walsh et al., 1998] W.E. Walsh, M.P. Wellman, P.R. Wurman, and J.R. MacKie-Mason. Some economics of market-based distributed scheduling. In Proceedings of 18th International Conference on Distributed Computing Systems, 1998.

[White and Pagurek, 1998] Tony White and Bernard Pagurek. Towards multi-swarm problem solving in networks. In Proceedings of Third International Conference on Multi-Agent Systems (ICMAS'98), pages 333-340, 1998.

[Wolfram, 1983] S. Wolfram. Stastical mechanics of cellular automata. Rev. Modern Physics, 55:601-644, 1983.

[Wolfram, 1984] S. Wolfram. Cellular automata as models of complexity. Nature, 311(4):419-424, 1984.

[Wolpert and Tumer, 1999] David Wolpert and Kagan Tumer. An introduction to collective intelligence. Technical Report NASA-ARC-IC-99-63, NASA Ames Research Center, 1999.

[Wyns, 1999] J. Wyns. Architecture for Holonic Manufacturing Systems - The Key to Support Evolution and Reconfiguration. PhD thesis, K.U.Leuven, PMA Division, 1999.

[Zlotkin and Rosenschein, 1994] Gilad Zlotkin and Jeffrey S. Rosenschein. Coalition, cryptography, and stability: Mechanisms for coalition formation in task oriented domains. In Proceedings of the 12th National Conference on Artificial Intelligence. Volume 1, pages 432-437, Menlo Park, CA, USA, July 31-August 4 1994. AAAI Press.

## Curriculum Vitae

| | |
|---|---|
| October 4, 1971 | Born in Bad-Saarow-Pieskow (Germany) |
| 1978 – 1988 | Primary School |
| 1988 – 1990 | Secondary School |
| 1990 – 1991 | Military Service |
| 1991 – 1997 | Student in Computer Science at Humboldt University Berlin |
| 1993 | "Vordiplom" (equiv. Bachelors Degree) in Computer Science |
| 1994 – 1995 | Thames Valley University, London (UK) – ERASMUS Inter-University Cooperation Program |
| 1995 – 1996 | Part-time Employee of PSI AG |
| March 1997 | Diploma in Computer Science (equiv. Masters Degree) |
| 1997 – 2000 | Employee of DaimlerChrysler AG, Research & Technology, Multi-Agent Systems Department and Ph.D. Student at Humboldt University (Computer Science) |
| March 2000 – | Member Technical Staff at Environmental Research Institute Michigan (ERIM), Ann Arbor, MI, USA |

**Eidestattliche Erklärung**

Hiermit erkläre ich, daß

- ich die vorliegende Dissertationsschrift „Return from the Ant – Synthetic Ecosystems for Manufacturing Control“ selbstständig und ohne unerlaubte Hilfe angefertigt habe;
- ich mich nicht bereits anderwärtig um einen Doktorgrad beworben habe oder einen solchen besitze;
- mir die Promotionsordnung der Mathematisch-Naturwissenschaftlichen Fakultät II der Humboldt-Universität zu Berlin bekannt ist.

Sven Brueckner, 16.05.2001